ICON 2015 TUTORIAL

ON

DEEP LEARNING and DISTRIBUTED REPRESENTATIONS

Presented By

Sudha Bhingardive

Rudramurthy V

Kevin Patel

Prerana Singhal

Under Direction of

Dr. Pushpak Bhattacharyya

Department of Computer Science and Engineering

IIT Bombay

Part I :: BASICS

Introduction To Deep Learning

Rudramurthy V

IIT Bombay

Plan

Introduction

- Traditional Machine Learning Algorithms

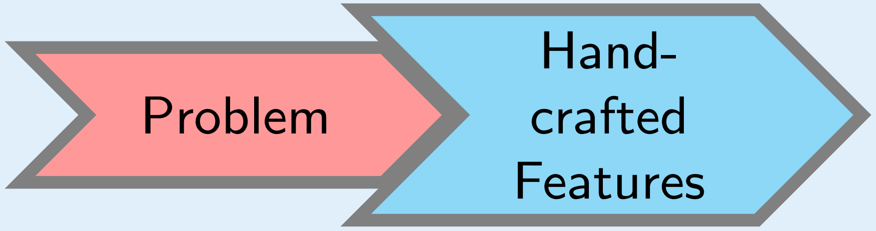
  Problem

- Deep Learning

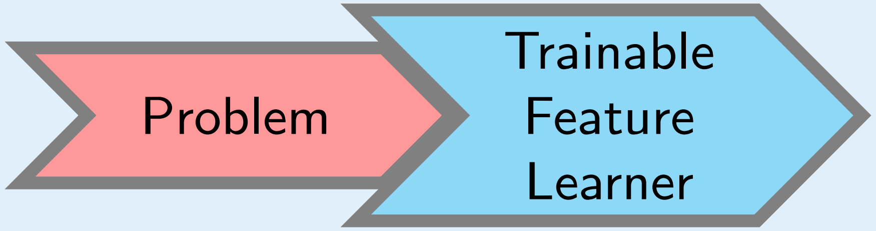
  Problem

Introduction

- Traditional Machine Learning Algorithms

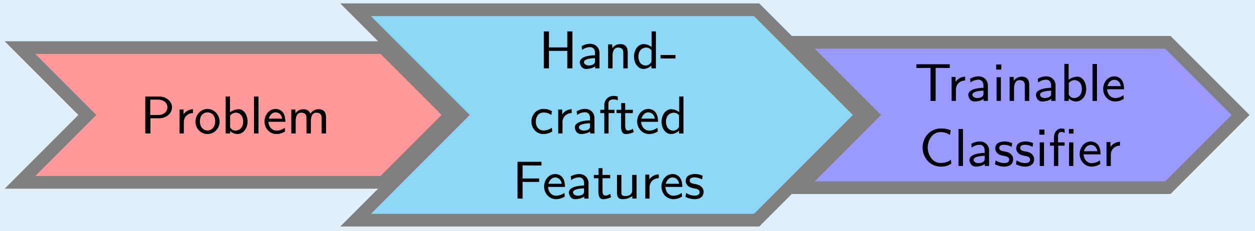


- Deep Learning

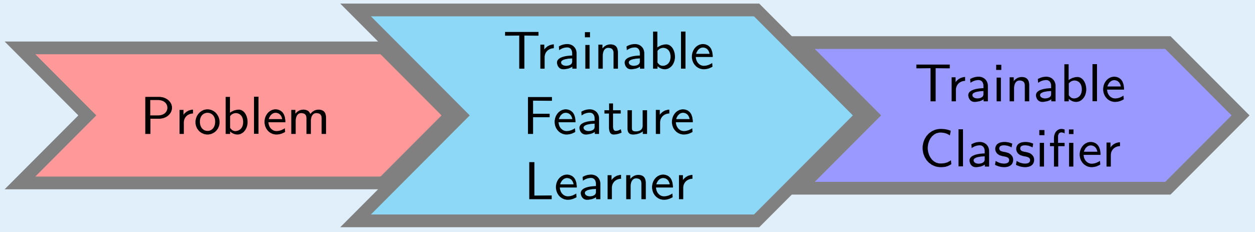


Introduction

- Traditional Machine Learning Algorithms

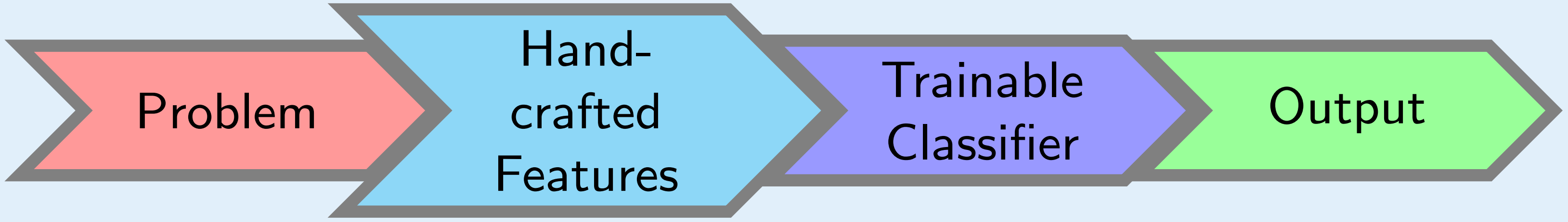


- Deep Learning

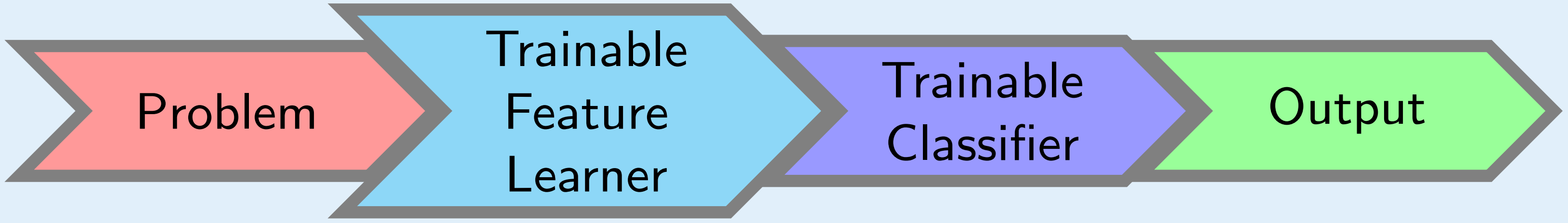


Introduction

- Traditional Machine Learning Algorithms



- Deep Learning



Part II :: APPLICATIONS

Named Entity Recognition & Deep Learning

Rudramurthy V

IIT Bombay

Plan

Named Entity Recognition

- Task of identifying entities in text
- Entities can be general like, **Person**,**Location**, **Organization**
- Or can be specific like **Medicine Name**, **Disease Name**
- Other entities could be **Date**, **Percent**,**Money**

Example:

- No sense in blaming the wicket - $(\text{Kohli})_{\text{Per}}$
- $(\text{Hilary Clinton})_{\text{Per}}$ is Hungry for War

Named Entity Recognition

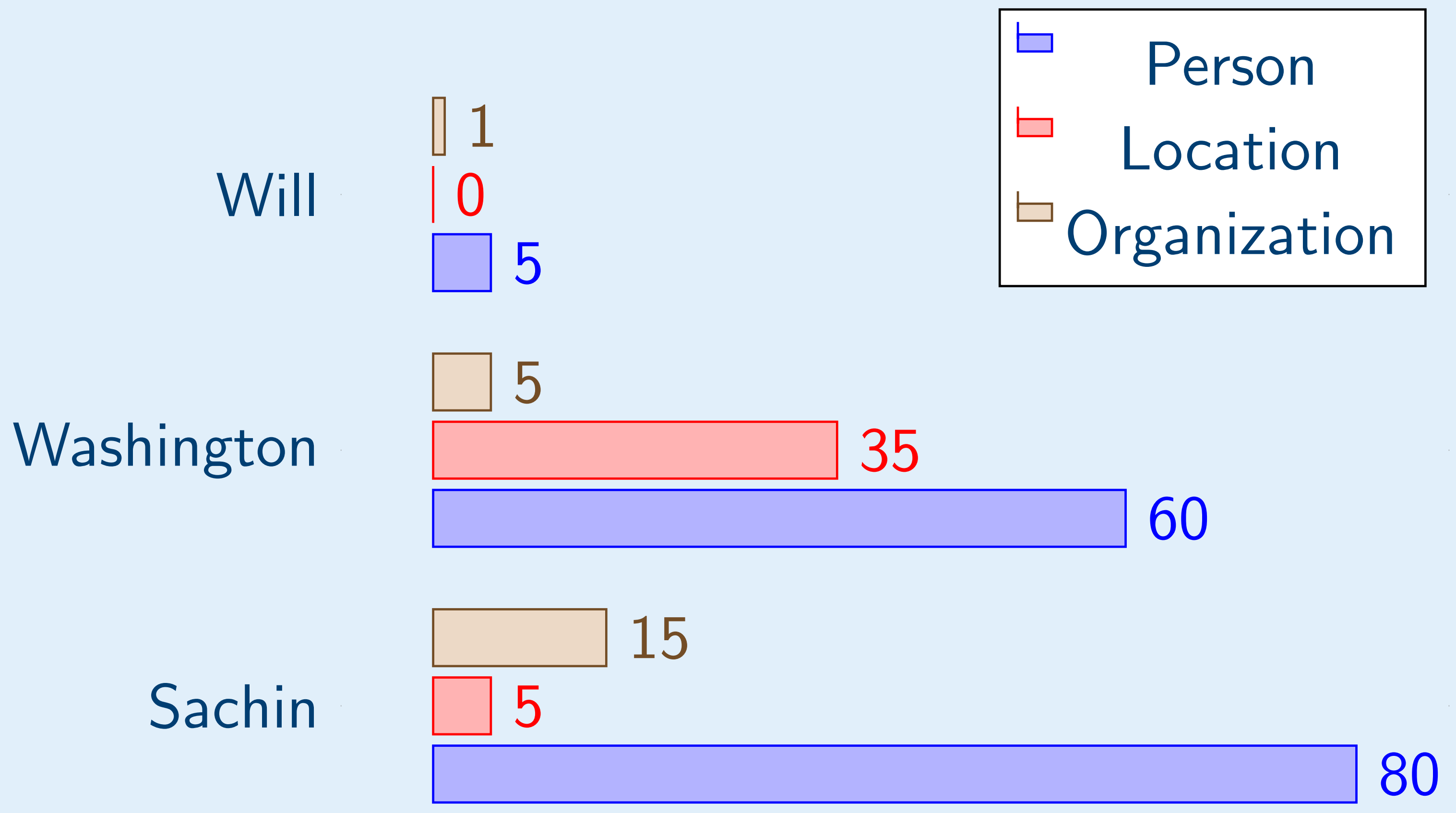
- Have a dictionary of Named Entities
- NER task would then become dictionary look-up
- Problem with ambiguous words
  - *Washington* as **Location** v/s *Washington* as **Person**
  - Unseen words (not present in training text)

Named Entity Recognition

- Define a distribution over words

Named Entity Recognition

Probability Distribution of Named Entity Tags for words



Named Entity Recognition

- Define a distribution over words
- Use context to predict ambiguous Named Entity tags
  - I went to $(\text{Washington})_{\text{Loc}}$ yesterday
  - I met $(\text{Washington})_{\text{Per}}$ yesterday
- Use language-specific features (Uppercase) to handle unseen words

Existing Systems

Features/Knowledge Resources used by Existing NER systems

- POS Tags
- Affixes
- Gazetteers
- Character level features like
  - Is First letter in Uppercase?
  - Contains digit?
  - Contains Non-alphanumeric characters?
  - ...
- ...

Word Embeddings [Pennington, Socher, and Manning 2014]

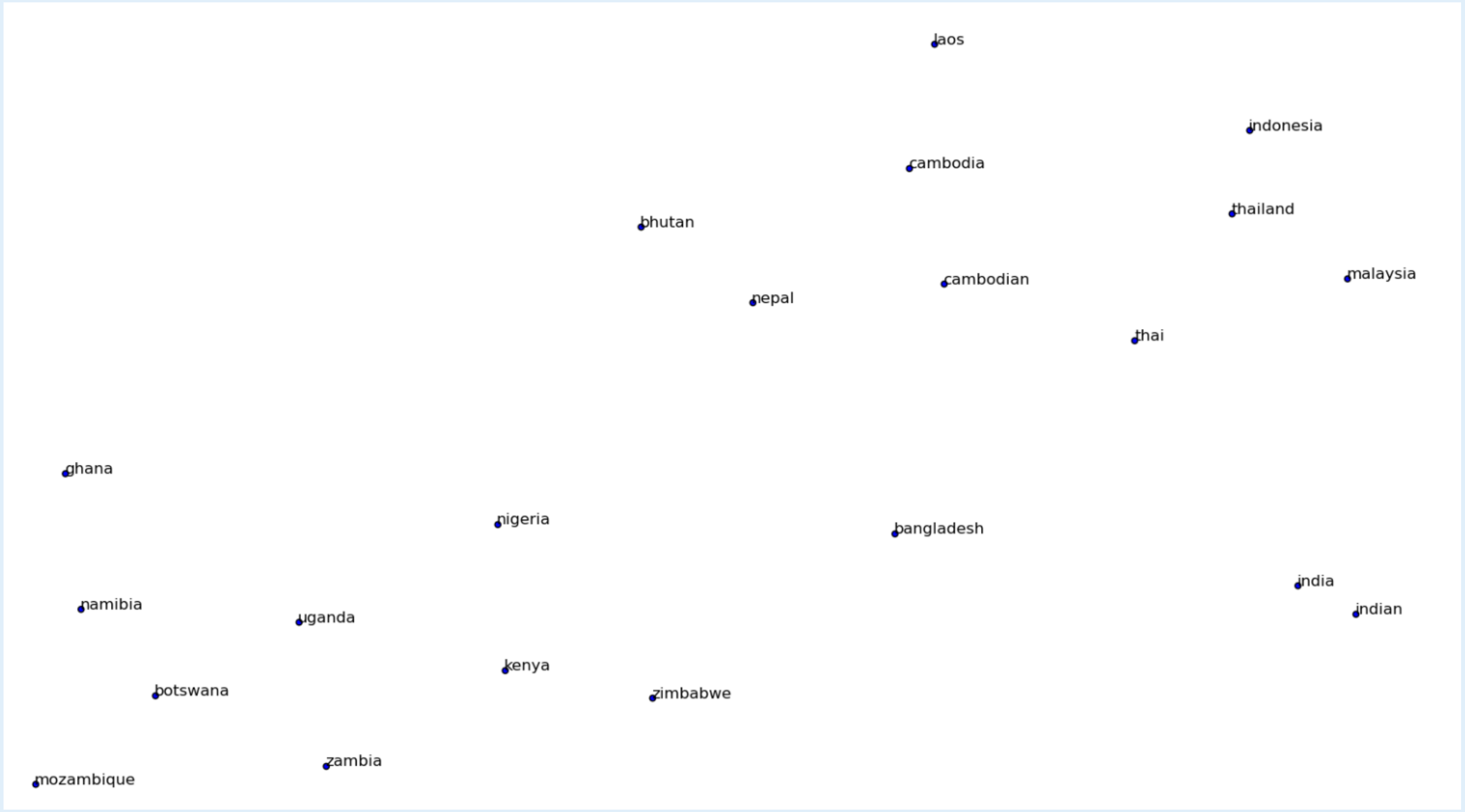


Figure: Projection of word embeddings from Glove in 2d

Word Embeddings [Mikolov, Yih, and Zweig 2013]

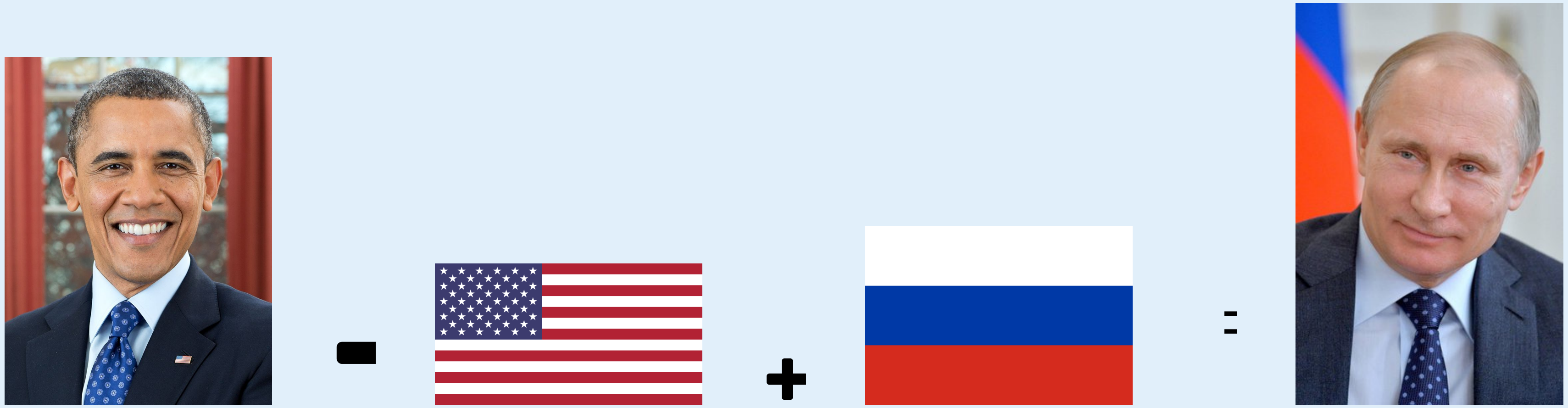


Figure: Obama - USA + Russia = Putin

Plan

Traditional Neural Network Approach for NER [Collobert et al. 2011]

- Words/unigram features are passed through a common layer
- The resulting features are then concatenated with other features like *uppercase*, *Gazetteers* ...
- The resulting features are then sent through a Neural Network for prediction

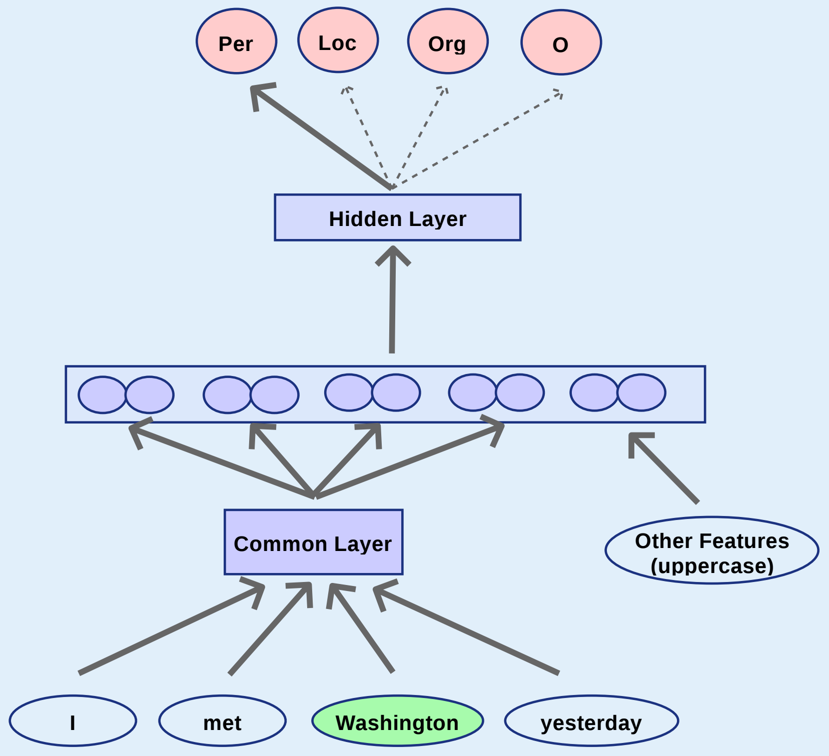


Figure: Traditional Neural Network Approach for NER

Experimental Setup

- Evaluation done on CoNLL 2003 NER Shared Task for English Tjong Kim Sang and De Meulder 2003
- Contains four tags, *Person*, *Location*, *Organization*, *Miscellaneous*
- Convert all words to lowercase
- Set *caps* feature if the word contains atleast one uppercase character
- Numbers are replaced by word *NUMBER*

Results [Collobert et al. 2011]

System	**F1 %**
Ando and Zhang 2005	89.31
Florian et al. 2003	88.76
Traditional Approach	79.53

Table: CoNLL English NER Shared Task Results

Unsupervised Representation Learning for Words [Collobert et al. 2011]

- Use approach similar to Autoencoder
- Given the context predict the middle word
- The features from common layer for every word acts as word embedding

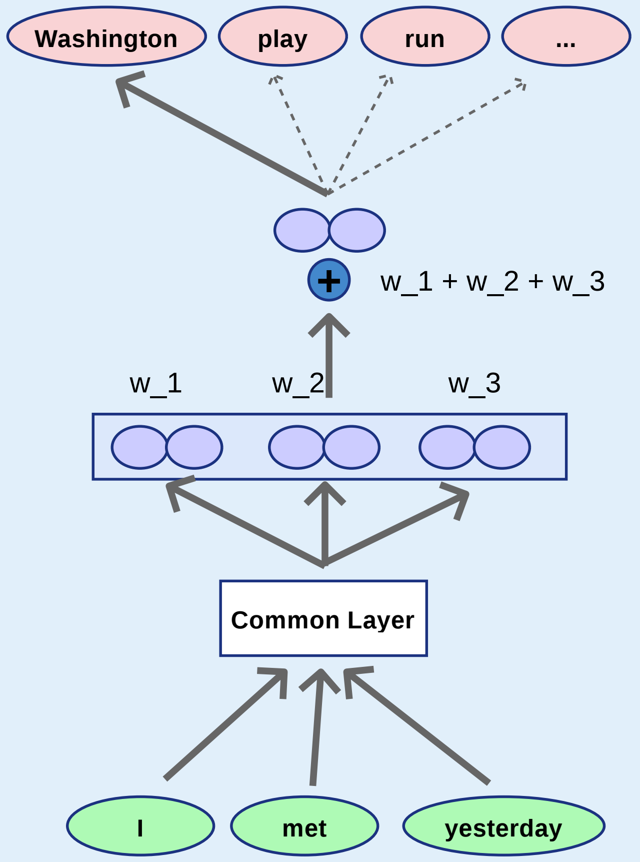


Figure: Representation Learning for words

Supervised Fine Tuning for NER [Collobert et al. 2011]

- Use the common layer from unsupervised learning to initialize the common layer for Neural NER
- Every word is passed through this common layer and corresponding features are extracted
- These features are then concatenated with other features like *Uppercase*, . . . and sent to higher layers for prediction
- The parameters of common layer are not updated during training

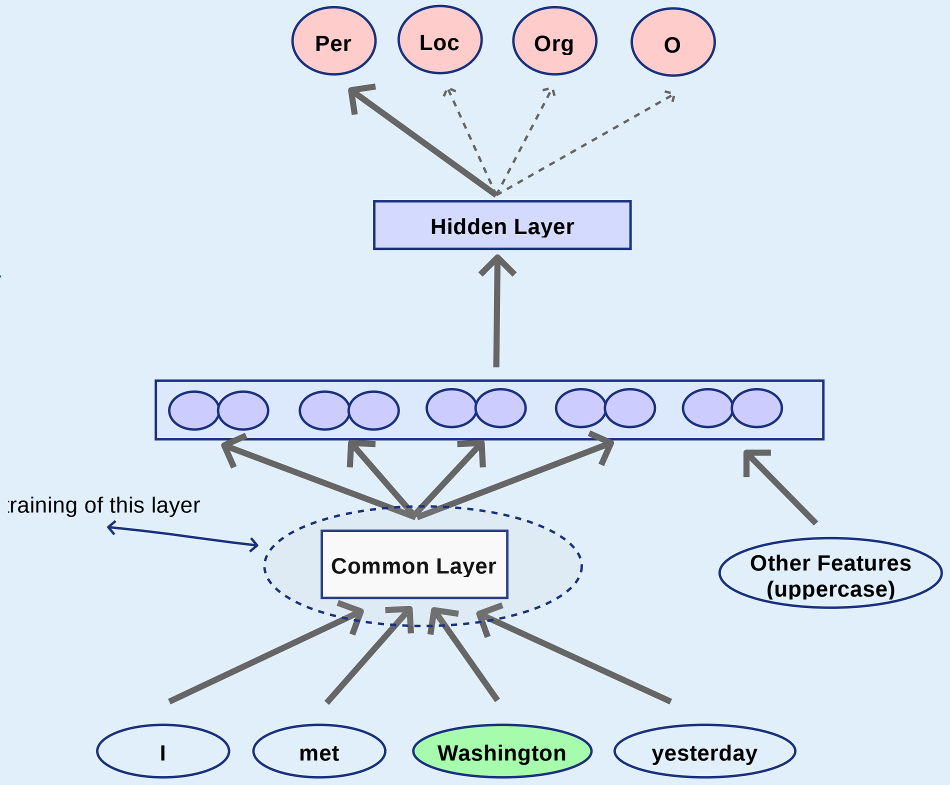


Figure: Feedforward Neural Network Architecture for NER

Word Level Log-Likelihood Results [Collobert et al. 2011]

System	**F1 %**
Ando and Zhang 2005	89.31
Florian et al. 2003	88.76
Traditional Approach	79.53
Traditional Approach + Unsupervised Pretraining	86.96

Table: CoNLL English NER Shared Task Results

Sentence-Level Log-likelihood (SLL) [Collobert et al. 2011]

- The earlier model only looked at the context and other features
- Information from Previous tag was not taken into consideration
- Some tags cannot follow other tags
- $(\textit{Mumbai University})_{\mathrm{ORG}}$

Sentence-Level Log-likelihood Architecture

- Obtain prediction for all words in a sentence using previous architecture
- Assuming a score for transitioning between tags
- Need to maximize the likelihood of taking valid path of tag sequence

Sentence-Level Log-likelihood Architecture

- The network outputs tag probabilities for every word
- The tag transition scores are represented by $f(\text{tag}_i \rightarrow \text{tag}_j) = A_{ij}$
- Score for sentence-tag sequence is given by

$$s([x]_1^N, [y]_1^n, \theta) = \sum_{i=1}^{N} \left(A_{[i]_{t-1}[i]_t} + P(y_i | x_t) \right)$$

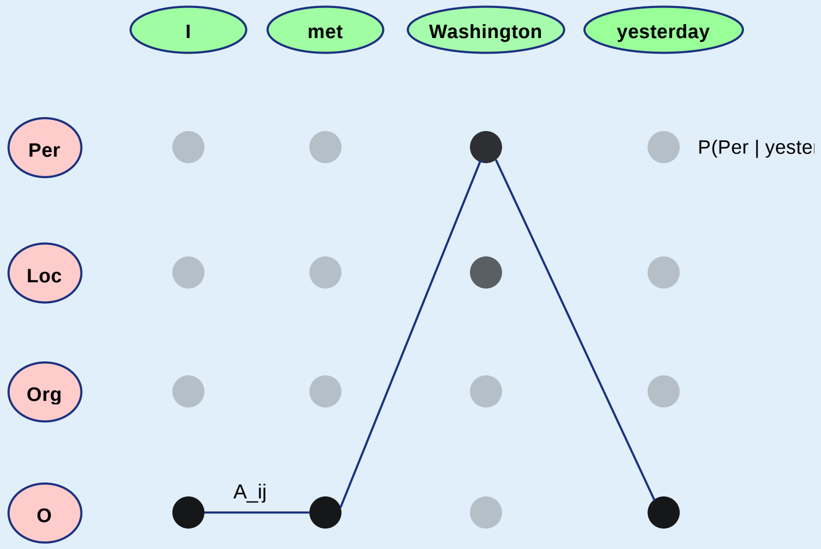


Figure: Feedforward Neural Network Architecture for NER

Sentence-Level Log-likelihood Architecture

- Score for sentence-tag sequence is given by

$$s([x]_1^N, [y]_1^n, \theta) = \sum_{i=1}^{N} \Big(A_{[i]_{t-1}[i]_t} + P(y_i | x_t) \Big)$$

- Maximize the score for valid tag sequence over all invalid tag sequences

$$\log P(s([x]_1^N, [y]_1^n, \theta) = s([x]_1^N, [y]_1^n, \theta) - \underset{\forall [j]_1^T}{\text{logadds}}([x]_1^N, [j]_1^n, \theta)$$

Sentence-Level Log-likelihood Architecture

- Training is done efficiently by using recursion to calculate the negative term
- Inference is done using Viterbi Algorithm

Sentence Level Likelihood Results

System	F1 %
Ando and Zhang 2005	89.31
Florian et al. 2003	88.76
Traditional Approach	79.53
Traditional Approach + Unsupervised Pretraining	86.96
Traditional Approach + Unsupervised Pretraining + SLL	88.67

Table: CoNLL English NER Shared Task Results

Character Convolutional Neural Network for NER [C. N. d. Santos and Guimares 2015]

- Previous approach still used some handcrafted features
- Can we make the system language independent by automatically learning these features?
- Presence of uppercase characters or digits or special symbols

Character Convolutional Neural Network for NER

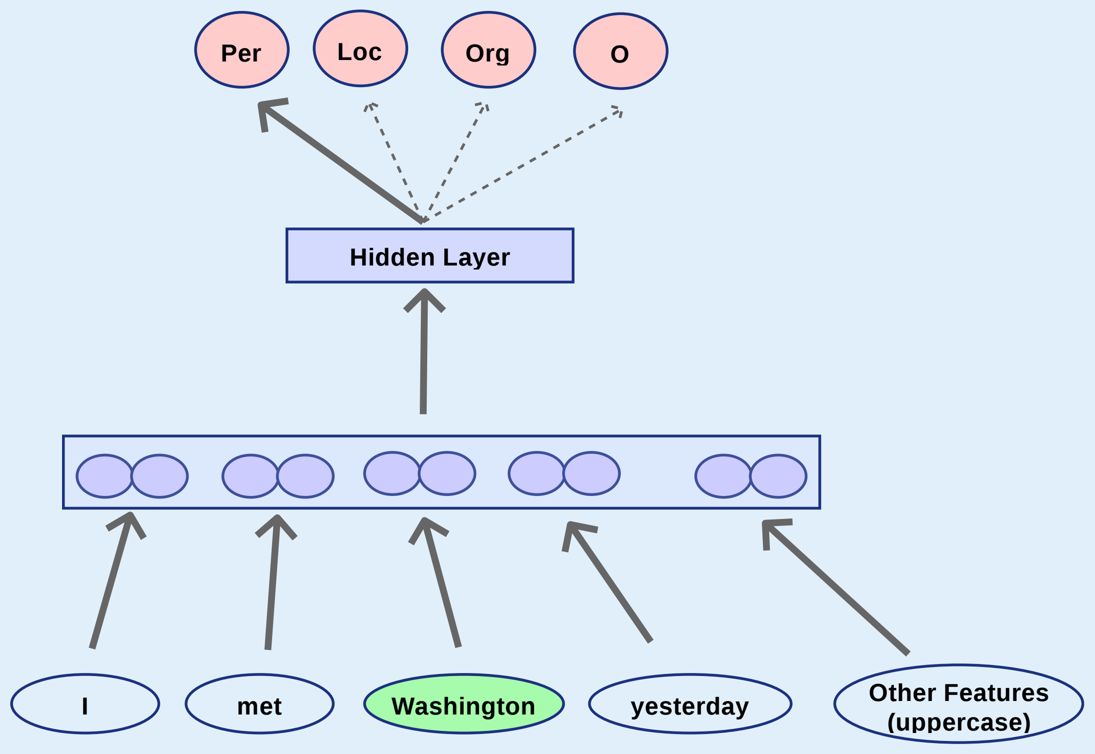


Figure: Feedforward Neural Network Architecture for NER with Handcrafted Features

Character Convolutional Neural Network for NER

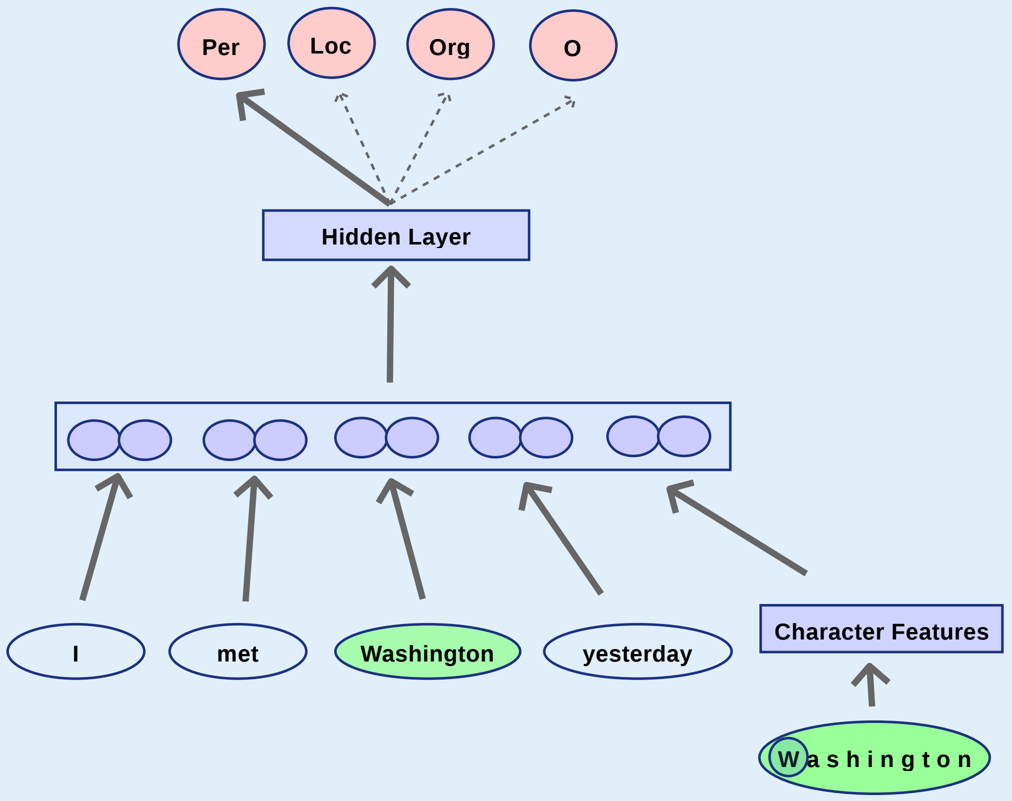


Figure: Feedforward Neural Network Architecture for NER with Learned Character Features

Character Convolutional Neural Network for NER

- Character representations are first extracted for every character
- Various feature detectors are applied across successive nGram characters
- Since we are interested in presence or absence of a feature
- Take maximum value for a particular feature across nGram characters
- These features are then concatenated with word embeddings

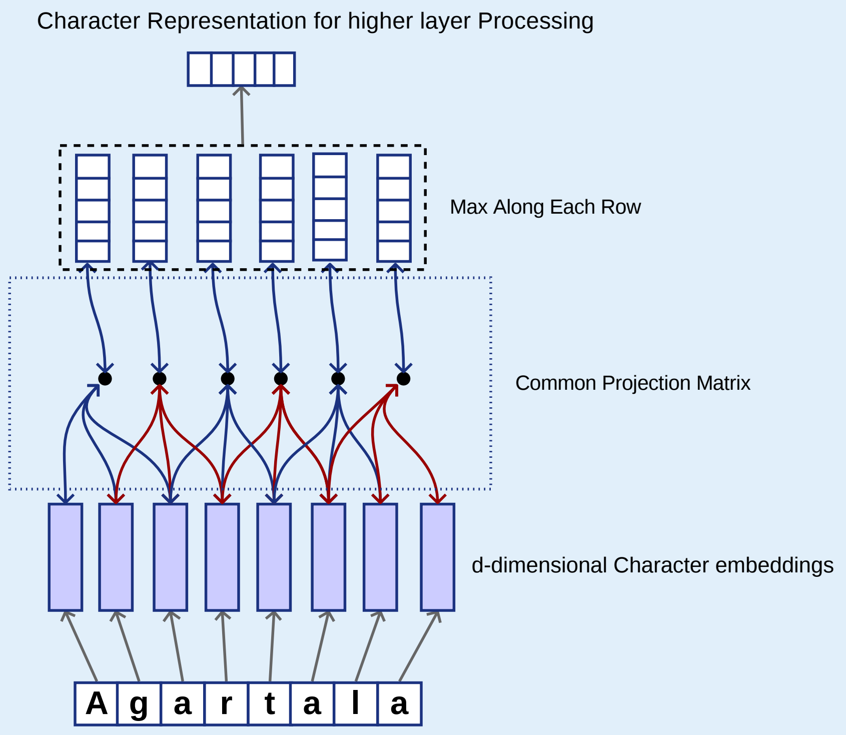

Character Representation for higher layer Processing
Max Along Each Row
Common Projection Matrix
d-dimensional Character embeddings
A g a r t a l a

Figure: Convolutional Neural Network to extract Character level Features

Character Convolutional Neural Network for NER

- The architecture used is similar to SENNA's architecture
- Character-level features are learned using Convolutional Neural Network
- Sentence-Level Log-Likelihood is used for training
- Inference is done using Viterbi Algorithm

Experimental Setup

- Experiments performed on Portuguese NER and Spanish NER
- Word embeddings were trained on respective Wikipedia corpus
- HAREM corpus [D. Santos and Cardoso 2006] was used for training and testing Portuguese NER
- CoNLL 2002 Spanish NER corpus [Tjong Kim Sang 2002] was used for training and testing purpose

Spanish NER: Results

System	**F1 %**
Carreras, Mrquez, and Padr 2003	81.39
CharWNN	82.21

Table: Comparison with the state-of-the-art for the SPA CoNLL-2002 corpus

Portuguese NER: Results

System	**F1 %**
ETL_{CMT} [C. d. Santos and Milidi 2012]	70.72
CharWNN	77.93

Table: Comparison with the State-of-the-art for the HAREM I corpus

Plan

Conclusion

- We have seen Language-independent NER using Deep Learning
- Word embeddings and character embeddings were employed for NER
- Results closer to state-of-the-art models were achieved
- Character level features trained from the training data were able to extract relevant character-level features for NER

References I

Rie Kubota Ando and Tong Zhang. “A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data”. In: *J. Mach. Learn. Res.* 6 (Dec. 2005), pp. 1817–1853. ISSN: 1532-4435. URL: `http://dl.acm.org/citation.cfm?id=1046920.1194905`.

Xavier Carreras, Llus Mrquez, and Llus Padr. “A Simple Named Entity Extractor Using AdaBoost”. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL ’03. Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 152–155. DOI: 10.3115/1119176.1119197. URL: `http://dx.doi.org/10.3115/1119176.1119197`.

References II

Ronan Collobert et al. "Natural Language Processing (Almost) from Scratch". In: *J. Mach. Learn. Res.* 12 (Nov. 2011), pp. 2493–2537. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1953048.2078186.

Radu Florian et al. "Named Entity Recognition Through Classifier Combination". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 168–171. DOI: 10.3115/1119176.1119201. URL: http://dx.doi.org/10.3115/1119176.1119201.

References III

- Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. “Linguistic Regularities in Continuous Space Word Representations”. In: *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*. 2013, pp. 746–751. URL: http://aclweb.org/anthology/N/N13/N13-1090.pdf.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “GloVe: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

References IV

Ccero Nogueira dos Santos and Victor Guimares. “Boosting Named Entity Recognition with Neural Character Embeddings”. In: *CoRR* abs/1505.05008 (2015). URL: `http://arxiv.org/abs/1505.05008`.

C.N. dos Santos and R.L. Milidi. *Entropy Guided Transformation Learning: Algorithms and Applications*. SpringerBriefs in Computer Science. Springer London, 2012. ISBN: 9781447129776. URL: `https://books.google.co.in/books?id=bptRcfoYFoMC`.

Diana Santos and Nuno Cardoso. *HAREM, a primeira avaliao conjunta de sistemas de reconhecimento de entidades mencionadas para portugu1^{e}s : documentaoeactasdoencontro*. Linguateca. 2006.

References V

Erik F. Tjong Kim Sang. “Introduction to the CoNLL-2002 Shared Task: Language-independent Named Entity Recognition”. In: *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*. COLING-02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 1–4. DOI: `10.3115/1118853.1118877`. URL: `http://dx.doi.org/10.3115/1118853.1118877`.

References VI

Erik F. Tjong Kim Sang and Fien De Meulder. “Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition”. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL ’03. Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 142–147. DOI: 10.3115/1119176.1119195. URL: http://dx.doi.org/10.3115/1119176.1119195.

Sentiment Analysis & Deep Learning

Prerana Singhal

IIT Bombay

Outline

- What is Sentiment Analysis ?
- Diversities Associated with Sentiment Analysis
- Why Deep Learning for Sentiment Analysis ?
- Deep Learning models for Sentiment Analysis
    - Convolution Neural Network
    - LSTM models
- Results obtained in Literature
- References

What is Sentiment Analysis ?

Non-trivial Field of study in NLP

Identify and analyse underlying opinion at:

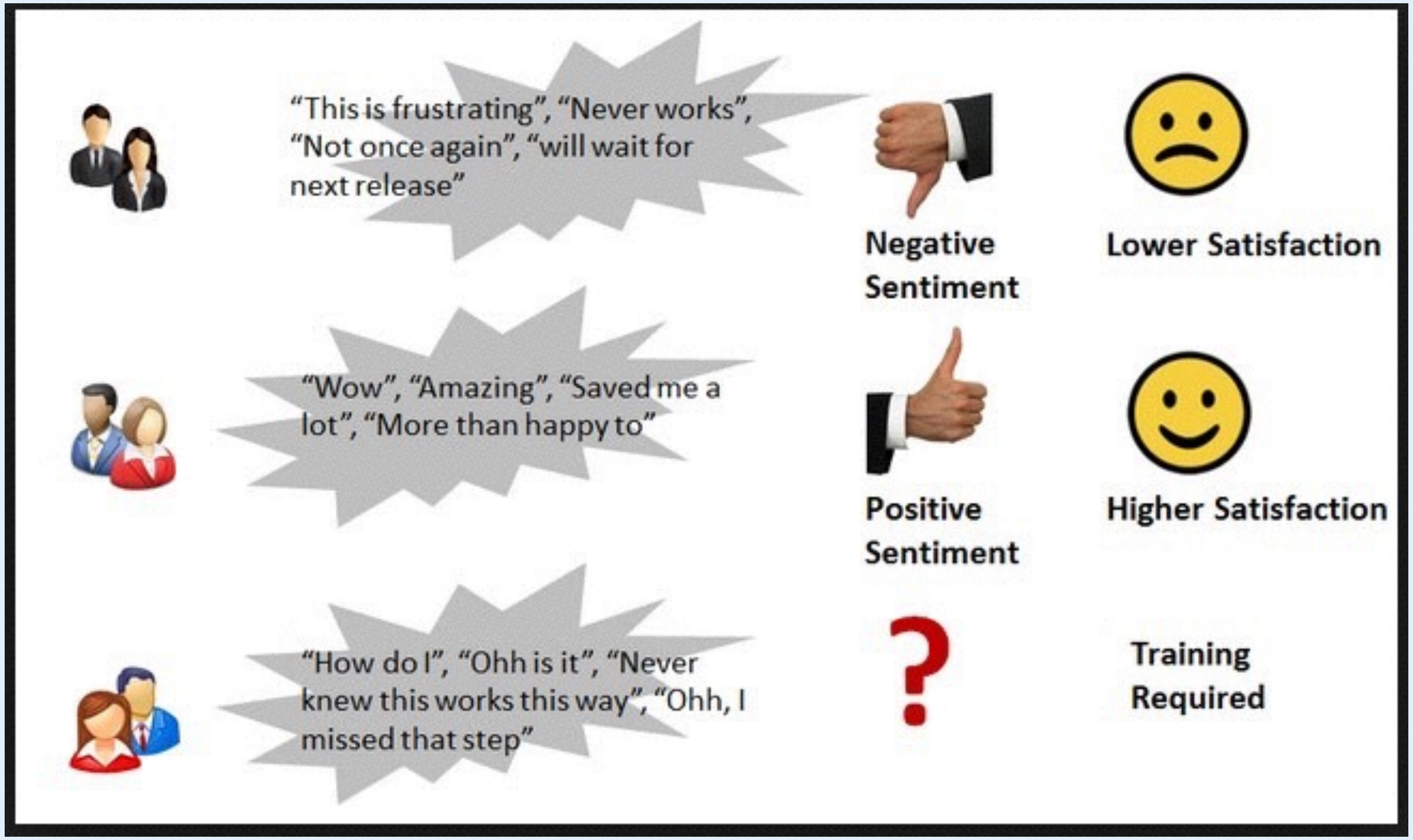
- ***document level***
- ***sentence level***
- ***entity/aspect level***

Identify positive/negative sentiments

Identify emotions

What is Sentiment Analysis ?

- **Non-Trivial because not at all straight-forward**
- **Even human beings have a hard time agreeing on the intended sentiment underlying a piece of opinion.**


"This is frustrating", "Never works", "Not once again", "will wait for next release"
Negative Sentiment
Lower Satisfaction
"Wow", "Amazing", "Saved me a lot", "More than happy to"
Positive Sentiment
Higher Satisfaction
"How do I", "Ohh is it", "Never knew this works this way", "Ohh, I missed that step"
?
Training Required

Diversities associated with Sentiment Analysis

Analysing sentiments from a given piece of text involves several aspects, each posing its own challenges

- Classification Outputs
- Type, Size and Domain of data
- Language
- Classification Type

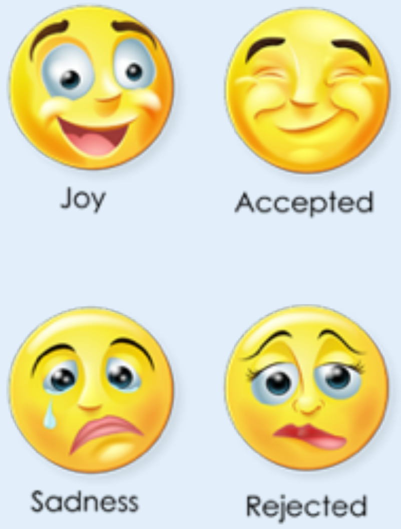
Diversities in Classification Output

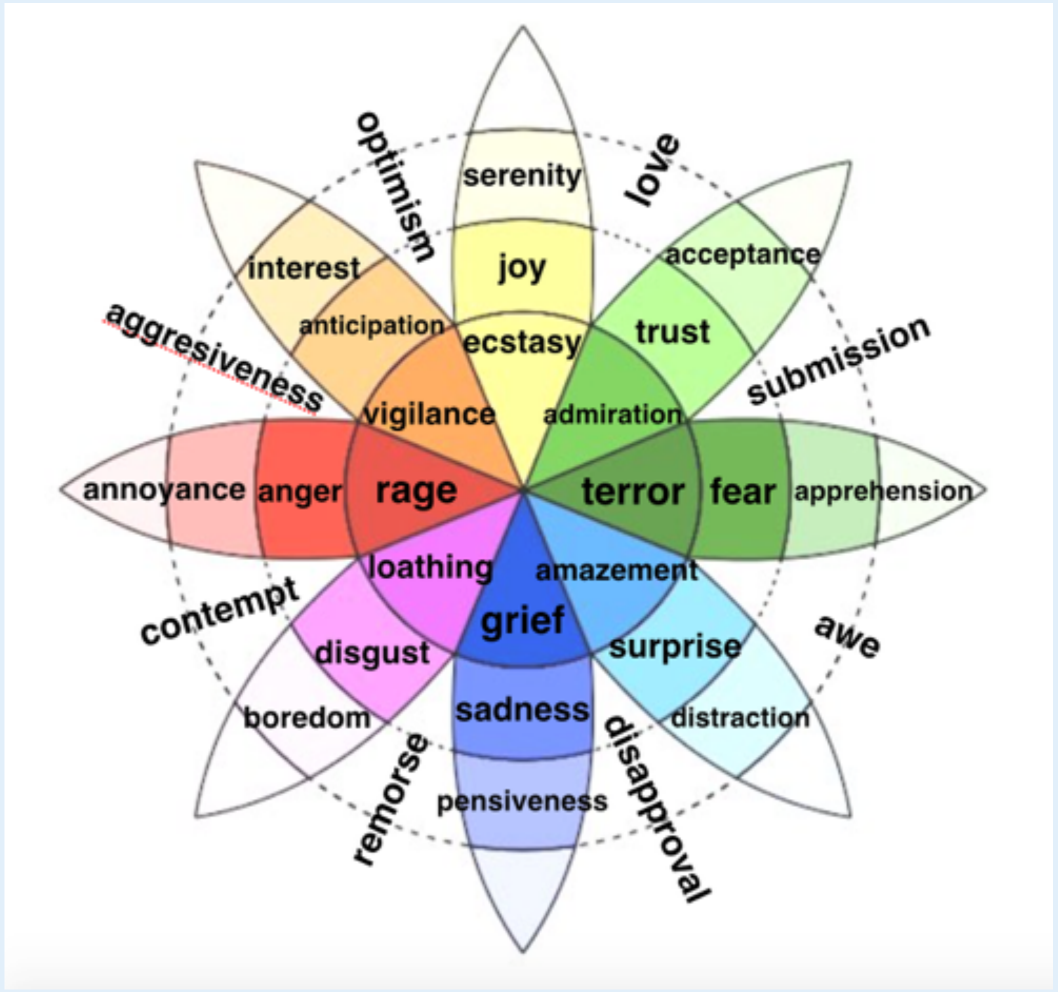
Crude Sentiments

Fine-Grained Sentiments

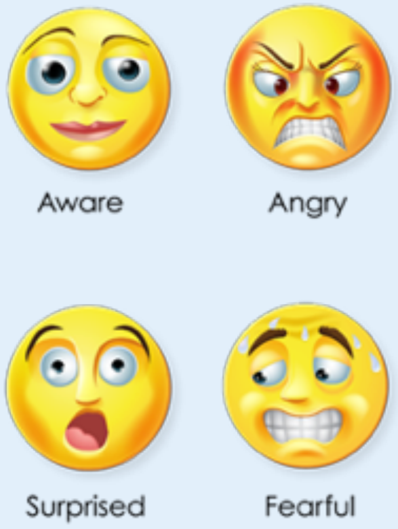
Diversities in Classification Output

Emotions





Plutchik's Wheel of Emotions



Diversities in Classification Output

- Identifying Sarcasm, Thwarting, etc.

  — Sarcatic / Non-sarcastic

  — Thwarted / Non-Thwarted

- Aspect Categorization

  — Aspects discussed in text (like food, genre, ambience, etc)

Diversities in Data

- Size of training data available
- Size of each piece of text
  - — Blogs :: large (hundreds of words)
  - — Reviews and mails :: medium (tens of words within 1/2 hundred)
  - — Tweets :: short (20-30 words)
- Language Model
  - — Blogs and Reviews :: well-constructed grammatically correct sentences
  - — Emails :: Grammatical and spelling inconsistencies
  - — Tweets and Facebook posts :: no language model, no grammar, no rules, and peculiarities like emoticons, hashtags, etc.
- Domain :: Specific (reviews on movies, tourism, etc.) or not specific (tweets, etc.)

Diversities in Languages

- Opinions do not owe themselves to any language
- Language is a medium of expressing opinions;

  — words need to convey meanings, not the other way round

- *Although the essence of the opinion does not owe its identity to any language;*

  *— the **intensity level may slightly vary** owing to the vocabulary available in the languages*

  — language models vary

- ***Cross-lingual help may not be available** due to lack of such corpus*
- *Dataset like those extracted from Twitter or Facebook may consist of **words from multiple languages***

Diversities in Classification Types

- **Multi-class Classification**
  - — Each data point is assigned to exactly one class
  - — Example :: Document-level or Sentence-level Sentiment Classification
- **Multi-label Multi-class Classification**
  - — Each data point can be assigned to more than one class
  - — Example :: Aspect Categorization
- **Aspect-based Sentiment Analysis**
  - — A complicated task
  - — First the aspects discuss in the text are to be identified, then the sentiments associated with each of the identified entities need to be figured

Why Deep Learning for Sentiment Analysis ?

- Existing Machine Learning algorithms give good results; however,

  — Manual feature engineering required :: difficult

  — System is heavily data-dependent and problem-dependent

  — Not much effective in cases (like tweets) which follow no language model

- Deep learning easily adaptable to the diversities of Sentiment Analysis with minimum human intervention

  — besides its other advantages already discussed earlier

Why Deep Learning for Sentiment Analysis ?

Diversities of sentiment analysis can be easily accommodated by deep neural neworks.

- **Classification Outputs** :: By changing the number of neurons in output layer, the network can learn based on training data
- **Data** :: Since no manual feature-engineering is done, the network is capable to learn from any kind of data, give enough amount and enough time
- **Languages** :: Since no language-specific properties are used, the neural networks can work effectively on texts belonging to language
- **Classification Types** :: Adjusting the activation functions and changes in hyper-parameters can easily accommodate the different types of classification. For instance,

  — Changing the output layer non-linearity from softmax to sigmoid for multi-label classification

  — Having two networks to serve as hierarchical classifiers for aspect-based sentiment analysis

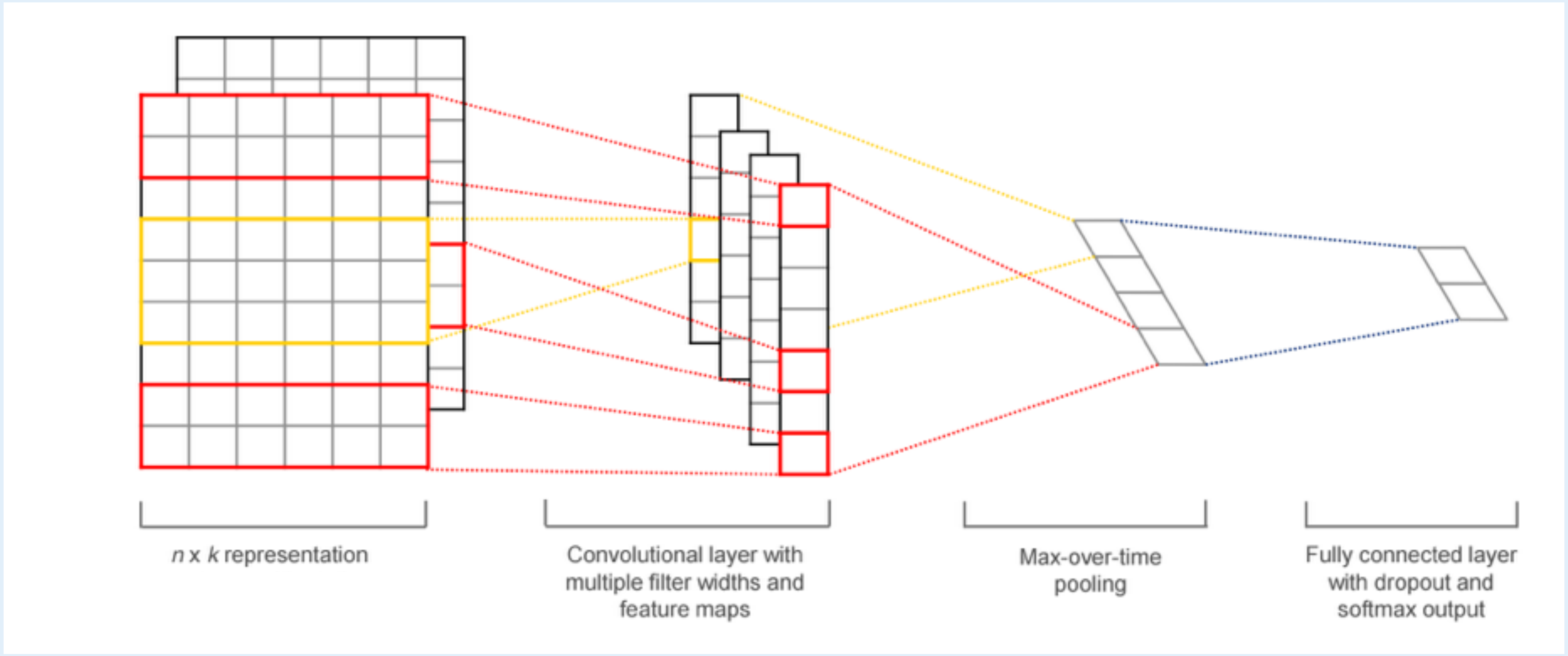
Deep Learning Models for Sentiment Analysis

- The words are transformed into feature vectors which are basically **word embeddings** learnt by training the neural network.

- *Window-approach network* is not desirable because many a times, classification w.r.t. to a particular word depends on some far-away word in the sentence not falling inside the window boundaries.

- Hence, sentence network approach is preferred for various NLP tasks including Sentiment Analysis.
  - — **Convolution Neural Network (CNN)**
  - — **Long Short-Term Memory Models (LSTM)**

CNN for sentiment analysis



n x *k* representation
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Model CNN architecture

Y. Kim, “Convolutional neural networks for sentence classification,” *arXiv preprint arXiv:1408.5882*, 2014.

CNN for sentiment analysis

3 versions ::

- **static** (word embeddings remain intact)
- **non-static** (word-embeddings change - learned by the network)
- **multichannel** (two channels - one static and one non-static)

Word embeddings used:

— trained on Google’s 300 billion news dataset.

— randomly initialised to be learned by the network

— sentiment-specific embeddings

CNN for sentiment analysis

Variant of CNN (based on Yoon Kim’s model) :: using **Dropout**

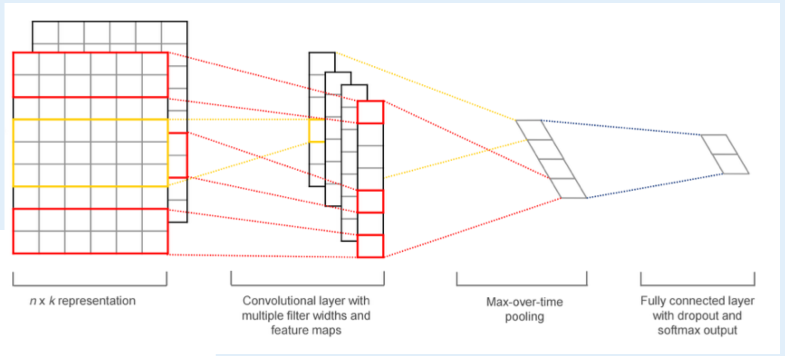
— The idea is to randomly mask/dropout/set to 0 some of the feature weights in each epoch (say a fraction of 0.4 of total number of neutrons)

— Prevents overfitting of training data to a large extent

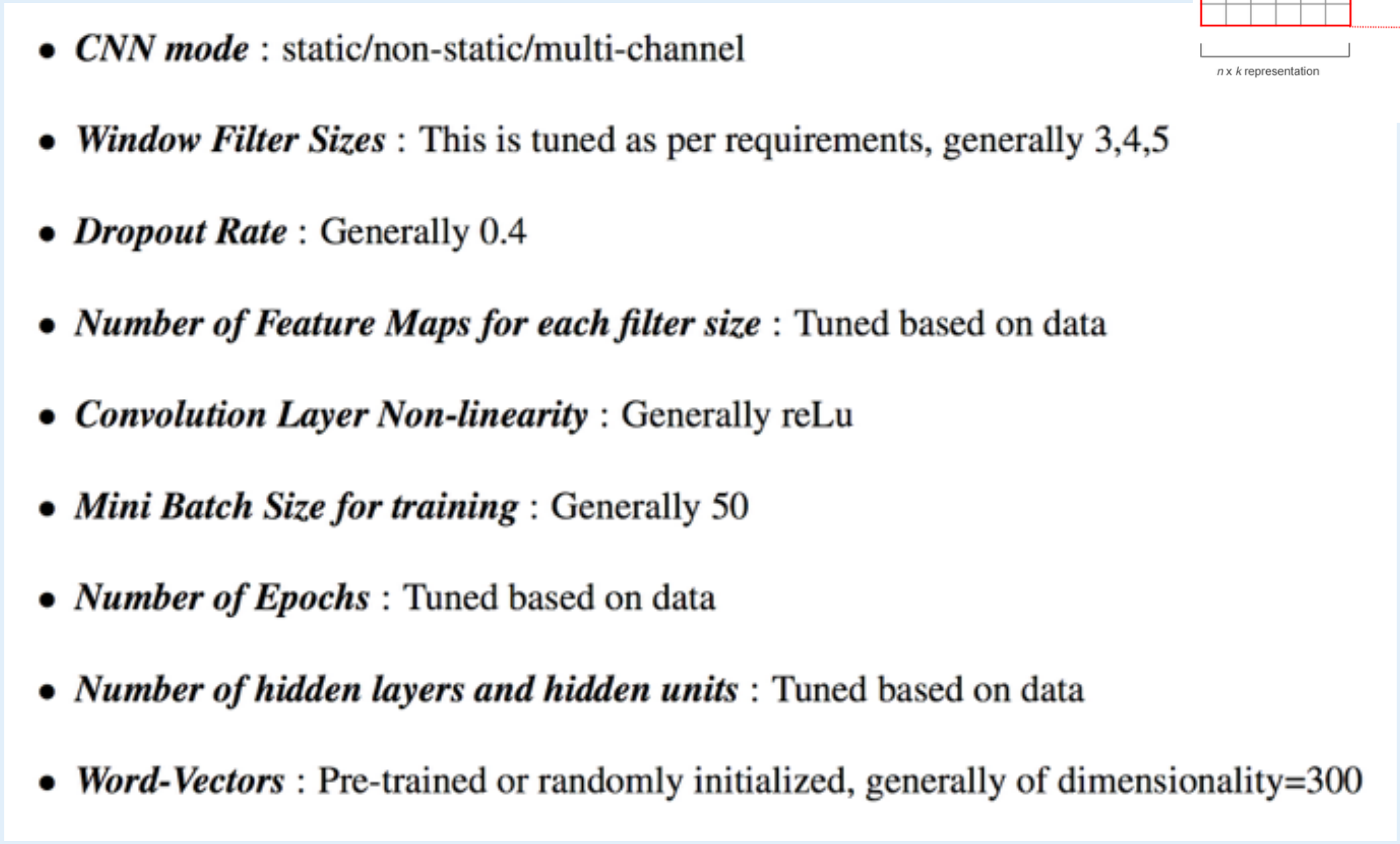
Y. Kim, “Convolutional neural networks for sentence classification,” *arXiv preprint arXiv:1408.5882*, 2014.

CNN for sentiment analysis

The basis of this model is CNN with dropout as suggested by Yoon Kim with the following hyper-parameters:



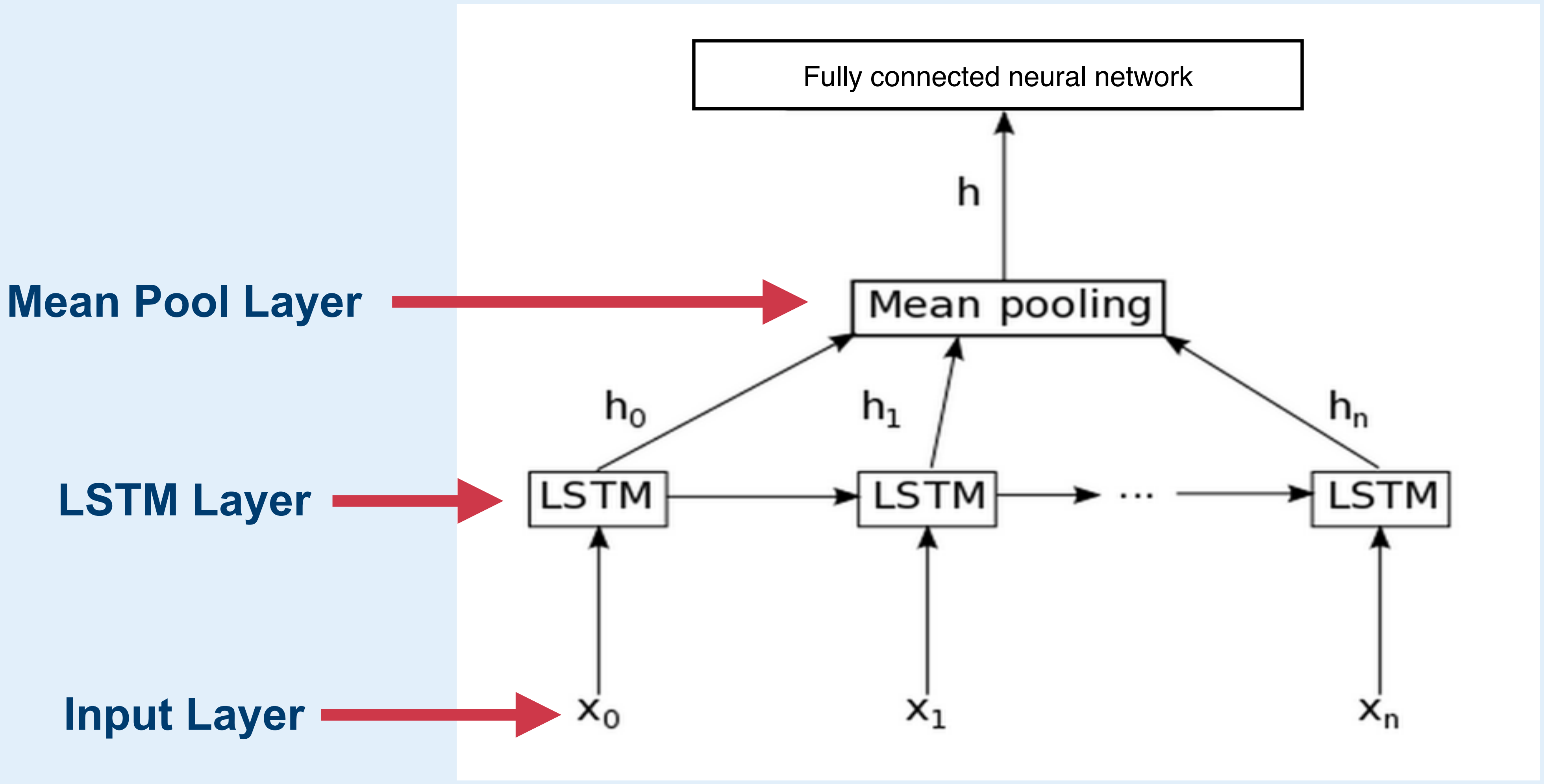
$n \times k$ representation
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output



- ***CNN mode*** : static/non-static/multi-channel
- ***Window Filter Sizes*** : This is tuned as per requirements, generally 3,4,5
- ***Dropout Rate*** : Generally 0.4
- ***Number of Feature Maps for each filter size*** : Tuned based on data
- ***Convolution Layer Non-linearity*** : Generally reLu
- ***Mini Batch Size for training*** : Generally 50
- ***Number of Epochs*** : Tuned based on data
- ***Number of hidden layers and hidden units*** : Tuned based on data
- ***Word-Vectors*** : Pre-trained or randomly initialized, generally of dimensionality=300

Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

LSTM for sentiment analysis

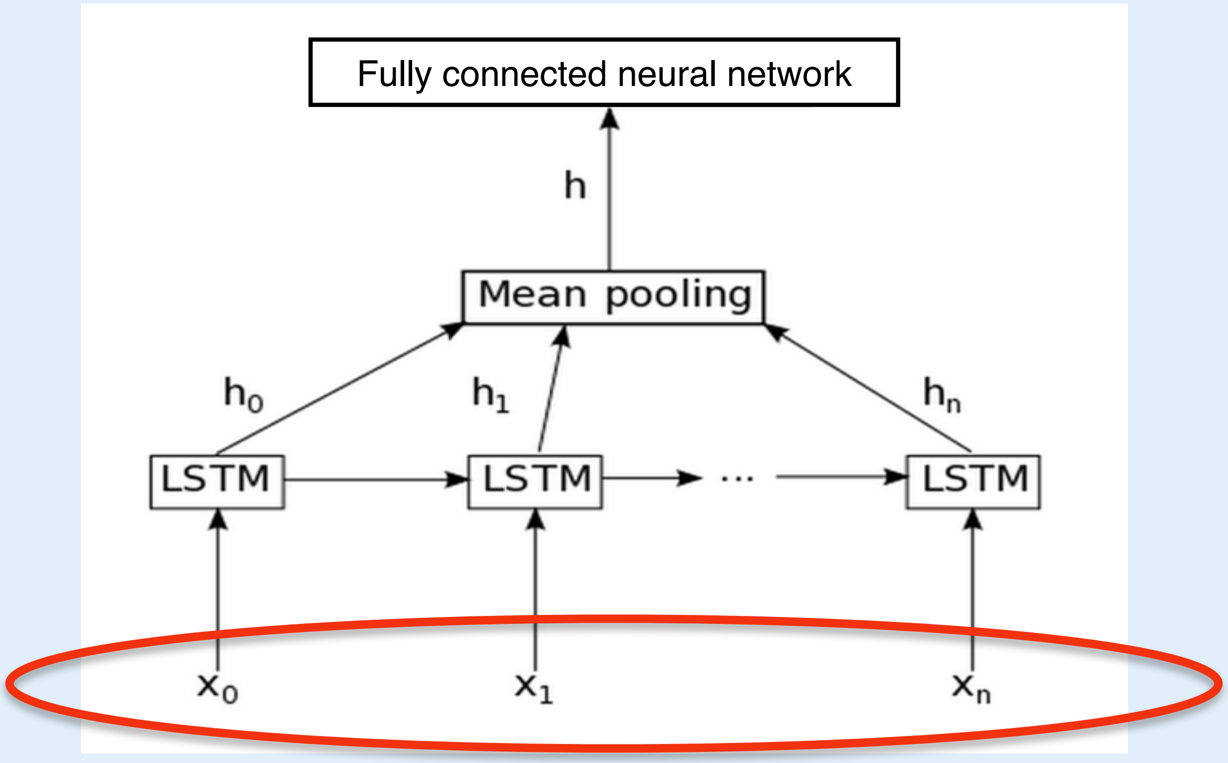


LSTM model for Classification Tasks

http://deeplearning.net/tutorial/lstm.html

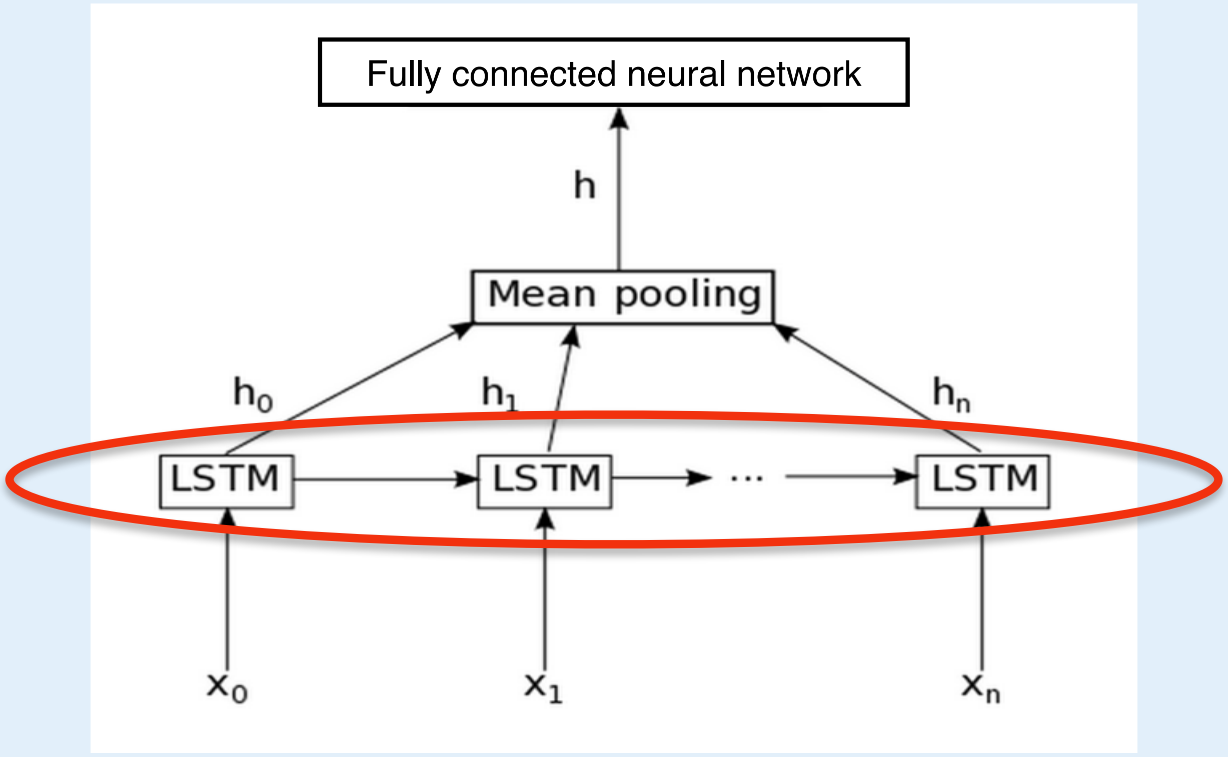
LSTM for sentiment analysis

- **Input layer**: This comprises of the feature vectors (of fixed dimensionality) of the words in the input sentence.
  - — Each time-step corresponds to one word in the sentence



LSTM for sentiment analysis

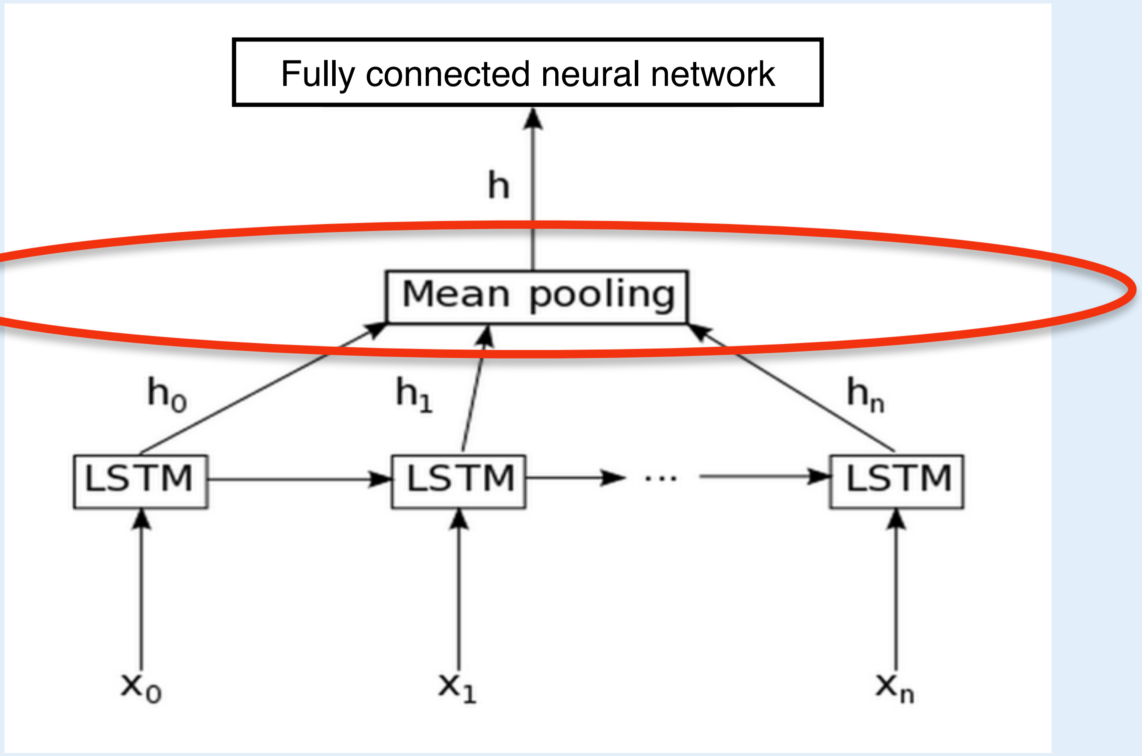
- **LSTM Layer**: A combination of channels in each of which the words are fed into the LSTM cell producing outputs at each time-step
  - These outputs correspond to the features of the whole sentence from the corresponding input word's point of view



LSTM for sentiment analysis

- **Mean Pool Layer**: Mean pooling operation is performed in each channel over the time-step outputs to average the sentence properties contained in the LSTM outputs.

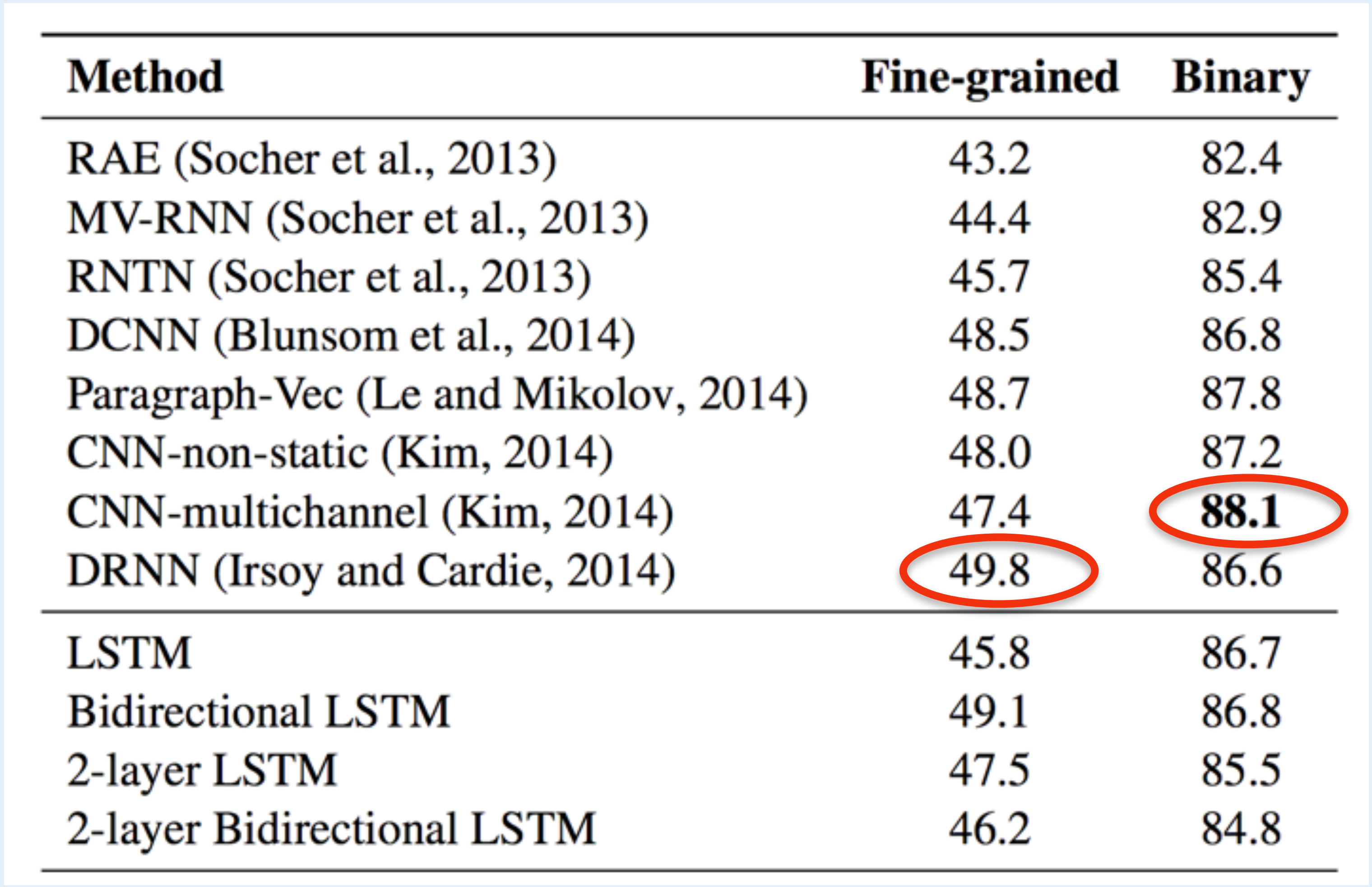
  — Averaging makes sense here as each value contains information of the whole sentence unlike in case of CNN where each value contained information of the neighbouring words in a window of pre-defined size.



LSTM for sentiment analysis

- The number of neurons in the output layer of the fully connected layer in the LSTM corresponds to the number of labels of classification.
- LSTM model does not use any window or phrases;
  — whole sentence works as input to the system
- Researchers have found this model to give appreciable results for various sentiment analysis tasks.

Results obtained

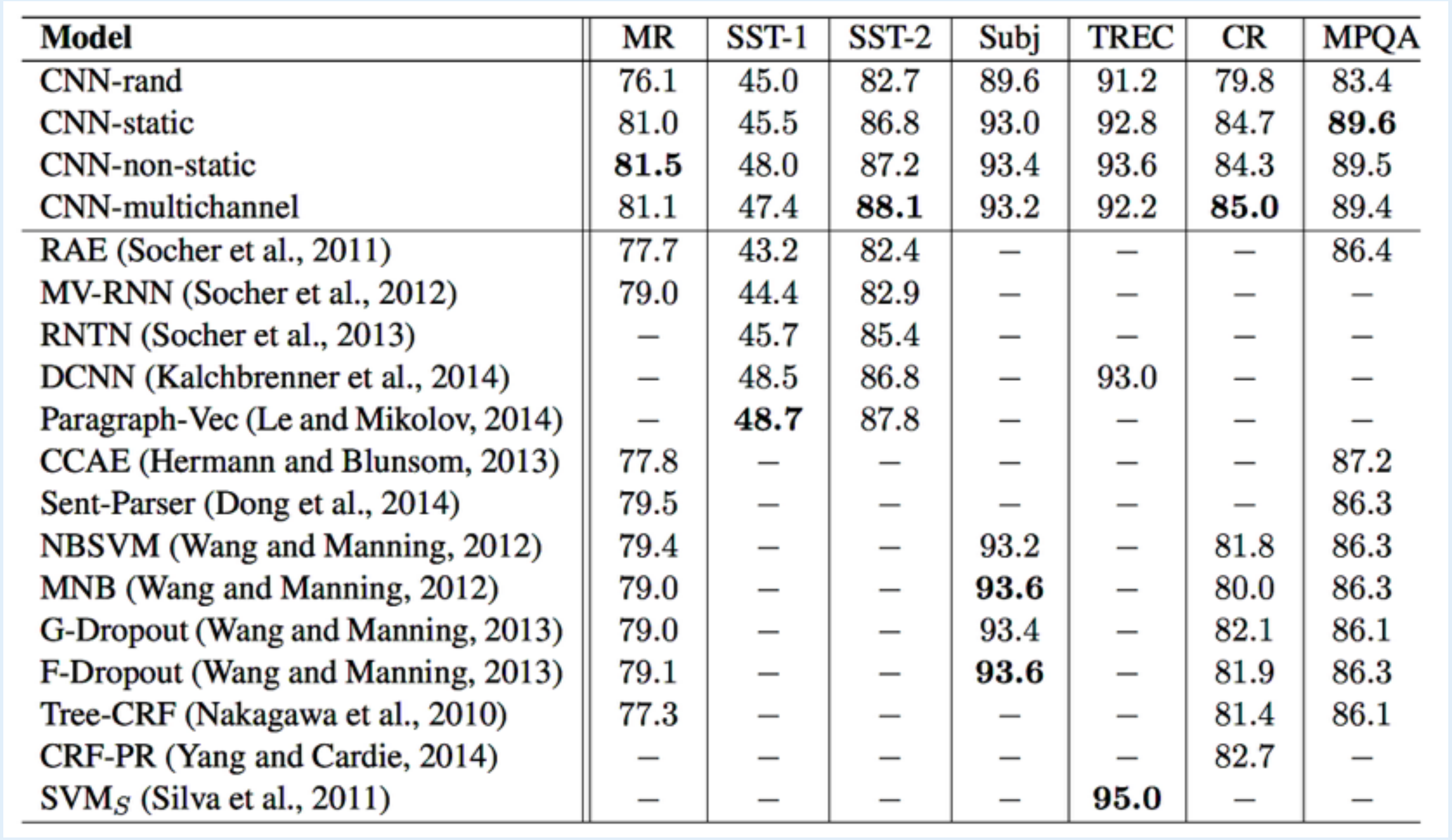


Method	Fine-grained	Binary
RAE (Socher et al., 2013)	43.2	82.4
MV-RNN (Socher et al., 2013)	44.4	82.9
RNTN (Socher et al., 2013)	45.7	85.4
DCNN (Blunsom et al., 2014)	48.5	86.8
Paragraph-Vec (Le and Mikolov, 2014)	48.7	87.8
CNN-non-static (Kim, 2014)	48.0	87.2
CNN-multichannel (Kim, 2014)	47.4	**88.1**
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
LSTM	45.8	86.7
Bidirectional LSTM	49.1	86.8
2-layer LSTM	47.5	85.5
2-layer Bidirectional LSTM	46.2	84.8

Test set accuracies on the Stanford Sentiment Treebank as reported in literature

Tai, Kai Sheng, Richard Socher, and Christopher D. Manning. "Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks." arXiv preprint arXiv:1503.00075 (2015).

Results obtained



Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Test set accuracies on different datasets by different models as reported in literature

Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).

Summary

— **Intersecting** the Natural Language Processing task of **Sentiment Analysis** with the complex problem-solving algorithm of **Deep Learning** seems to be a good idea.

— The deep learning approach promises one thing : **given sufficient amount of data and sufficient amount of training time**, it can perform the task of sentiment classification on any text, with **no restriction** on language, language model, or domain.

— A key asset of deep learning - **Adaptability** to the diverse problem statements falling under sentiment classification

— The **results** obtained by applying neural network models to different datasets look **promising**.

References

- Y. Kim, "**Convolutional neural networks for sentence classification**," *arXiv preprint arXiv: 1408.5882*, 2014.
- R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "**Natural language processing (almost) from scratch**," *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.
- S. Hochreiter and J. Schmidhuber, "**Long short-term memory**," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- "**Lstm networks for sentiment analysis**." *http://deeplearning.net/tutorial/ lstm.html.*
- **"Convolutional Neural Networks (LeNet)"** *http://deeplearning.net/tutorial/lenet.html*
- _"**Deeply moving: Deep learning for sentiment analysis.**" *http://nlp.stanford. edu/sentiment/*.
- A. Balamurali, "**Cross-lingual sentiment analysis for indian languages using linked wordnets,**" 2012.
- A. Joshi, A. Balamurali, and P. Bhattacharyya, "**A fall-back strategy for sentiment analysis in hindi: a case study**," *Proceedings of the 8th ICON*, 2010.
- Bing Liu, **Sentiment Analysis and Opinion Mining**. 2012.
- R. M. Richard Socher, Milad Mohammadi, "*Deep learning for nlp*." *http://cs224d.* stanford.edu/ lecture_notes/LectureNotes4.pdf
- "**Word2vec**." https://code.google.com/p/word2vec/.

Word Sense Disambiguation & Deep Learning

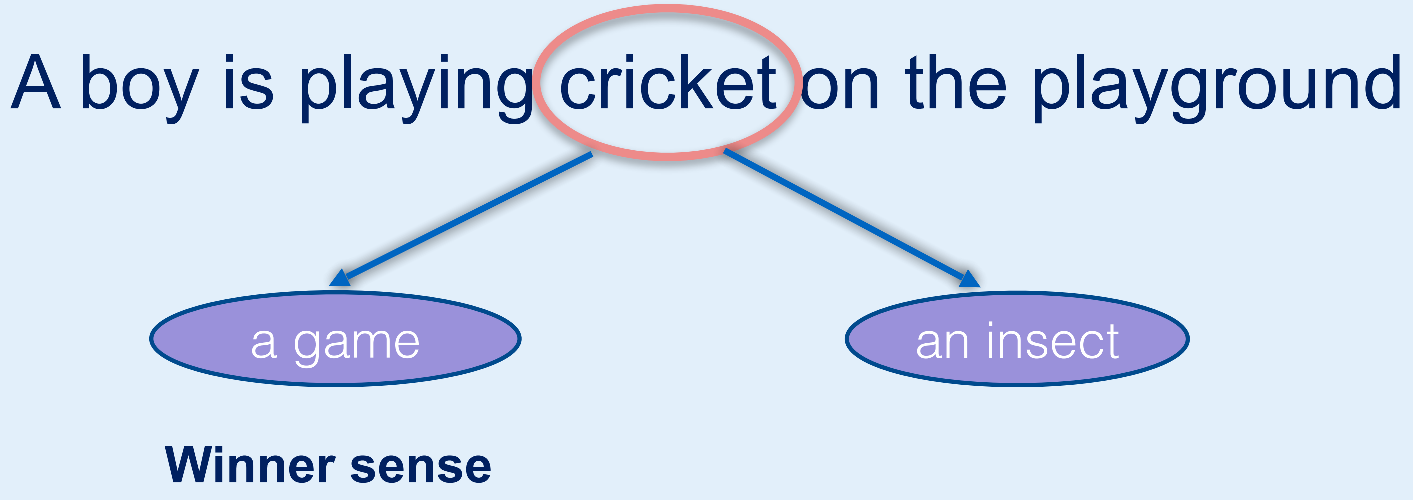
Sudha Bhingardive

IIT Bombay

Outline

- What is Word Sense Disambiguation ?
- Different approaches of WSD
- Most Frequent Sense
- MFS using Deep Learning
- Experiments and Results
- Summary
- References

What is Word Sense Disambiguation?

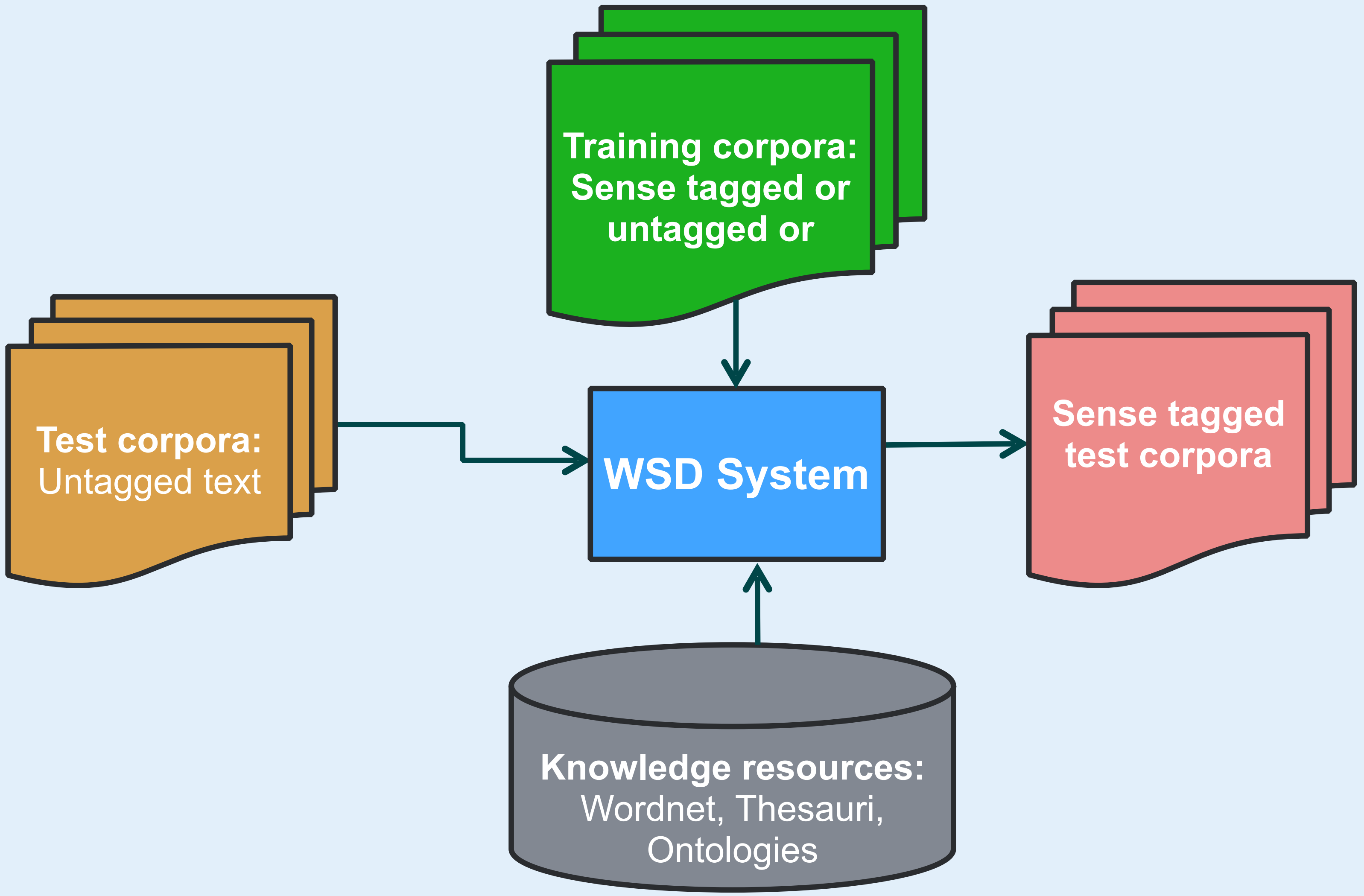


WSD is defined as identifying the meaning of words in a particular context

Different approaches of WSD

- Supervised approaches
  - Rely on sense-annotated corpus
  - Show very good performance
- Unsupervised approaches
  - Do not rely on sense-annotated corpus
  - Accuracy is less than supervised approaches
- Knowledge based approaches
  - Rely on the quality of the knowledge resources
  - Accuracy is less than supervised approaches

Block diagram of WSD



What is Most Frequent Sense WSD?

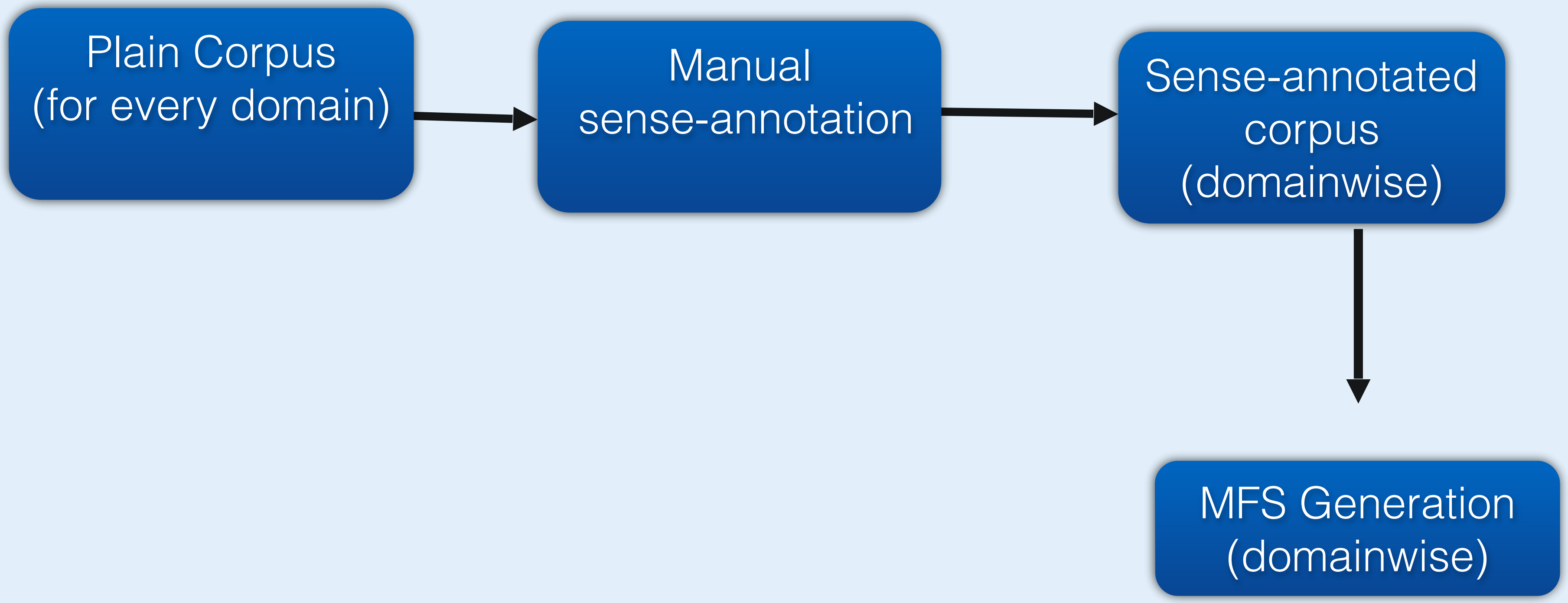
- Assigns the most frequent sense to every content words in the corpus
- Context is not considered
- For example: cricket [S_1 : game sense S_2: insect sense]
- If MFS (cricket) = S_1

1. A boy is playing cricket_S_1 on the playground
2. Cricket_S_1 bites won't hurt you
3. Cricket_S_1 singing in the home is a sign of good luck

Why Most Frequent Sense WSD?

- Strongest baseline in WSD
- Heuristic of choosing the most common sense is extremely powerful
- An acid test for any new WSD algorithm is its performance against the MFS
- Unsupervised WSD approaches generally fail to beat this baseline

How MFS baseline is created?


Plain Corpus (for every domain)
Manual sense-annotation
Sense-annotated corpus (domainwise)
MFS Generation (domainwise)

MFS using Deep Learning

- An unsupervised approach for MFS detection using word embeddings (Bhingardive et al., 2015)
- Training do not require any sense-annotated-corpora for training
- *Word embedding of a word is compared with all sense embeddings and obtain the predominant sense with the highest similarity.*
- Domain independent approach and can be easily ported across multiple languages

Word Embeddings

- Vector representation of word
- Represent each word with low-dimensional real valued vector.
- Increasingly being used in variety of Natural Language Processing tasks,
- **word2vec** tool (Mikolov et al, 2013)
  - takes a corpus as input and produces word vectors as output
  - first constructs a vocabulary from training data and learns vector representation of words
  - It captures many linguistic regularities
  - Vector(‘King’) –Vector(‘man’)+Vector[‘women’]=>Vector(‘queen’)
  - Work under the assumption that similar words occur in similar context

Word Embeddings contd..

- Related words given by word2vec tool for **cricket**
- Word: **cricket** Position in vocabulary: 3941

Word	Cosine distance
cricketing	0.837223
cricketers	0.816575
Test_cricket	0.809482
Twenty##_cricket	0.806849
Twenty##	0.762427
Cricket	0.754140
cricketer	0.737258
twenty##	0.731635
T##_cricket	0.730462

Looking at the top related words we can see Word Embeddings help in capturing MFS of words

Sense Embeddings

- Sense embeddings are obtained by taking the average of word embeddings of each word in the sense-bag

$$vec(S_i) = \frac{\sum_{x \in SB(S_i)} vec(x)}{N}$$

- S_i - ith sense of a word W
- N - Number of words present in the sense-bag $SB(S_i)$
- The sense-bag for the sense S_i is created as below,

$$\mathrm{SB}(S_i)=\{\mathrm{x}|\mathrm{x}\text{ - }\mathrm{Features}(S_i)\}$$

- Features(S_i) - WordNet based features for sense S_i

WordNet Features

- English WordNet entry for a word “mind” :
- Synset*:*

{**05619057**} <noun.cognition>: (n) mind, head, brain, psyche, nous (that which is responsible for one's thoughts, feelings, and conscious brain functions; the seat of the faculty of reason) *"his mind wandered"; "I couldn't get his words out of my head“*

- Synset ID: 05619057
- Synset Members: mind, head, brain, psyche, nous
- Gloss definition: that which is responsible for one's thoughts, feelings, and conscious brain functions; the seat of the faculty of reason
- Example Sentences: "his mind wandered"; "I couldn't get his words out of my head“

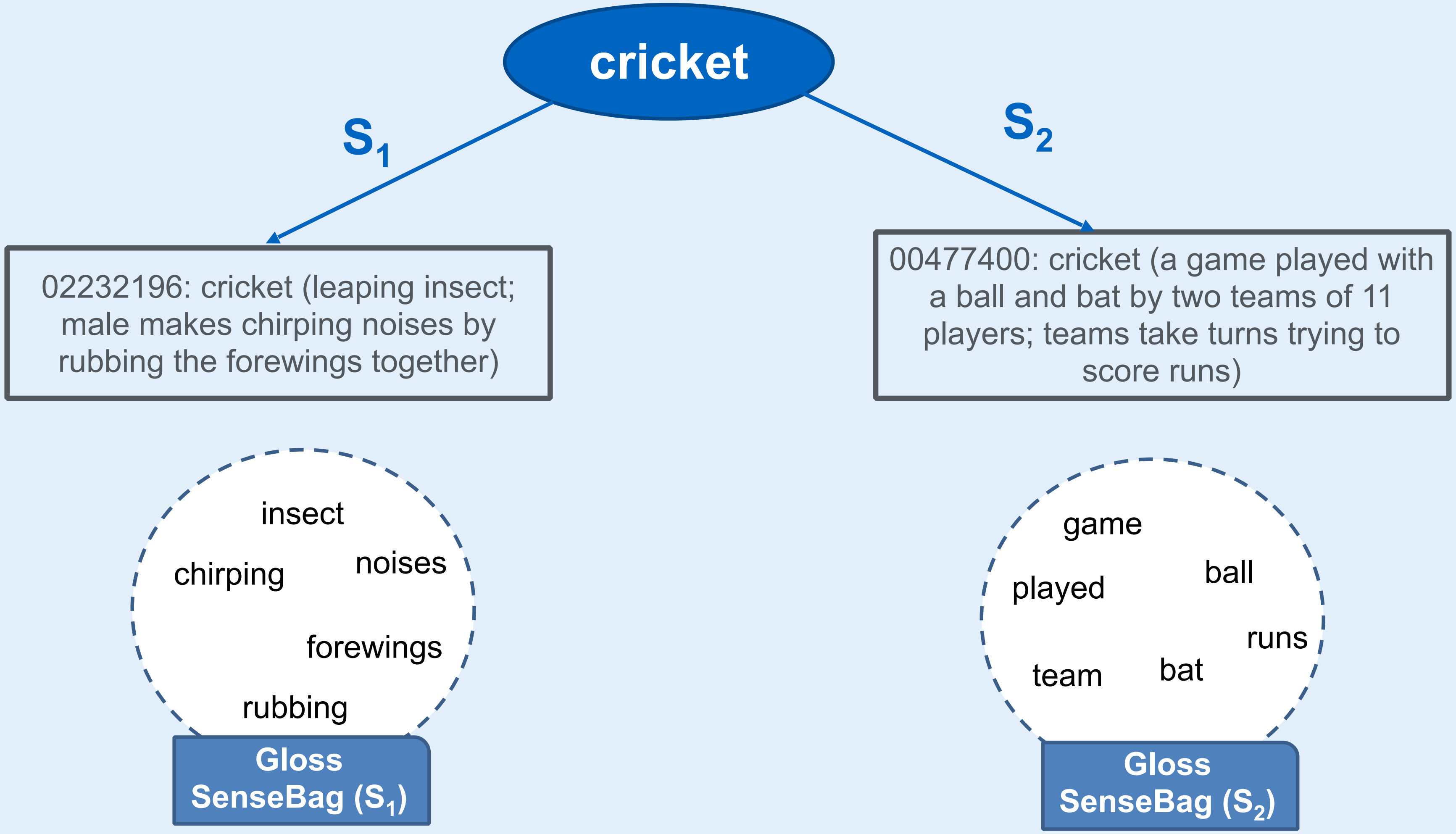
MFS Detection

- MFS identification problem is treated as finding the closest cluster centroid (*i.e.*, sense embedding) with respect to a given word.
- Cosine similarity is used.
- Most frequent sense is obtained by using the following formulation,

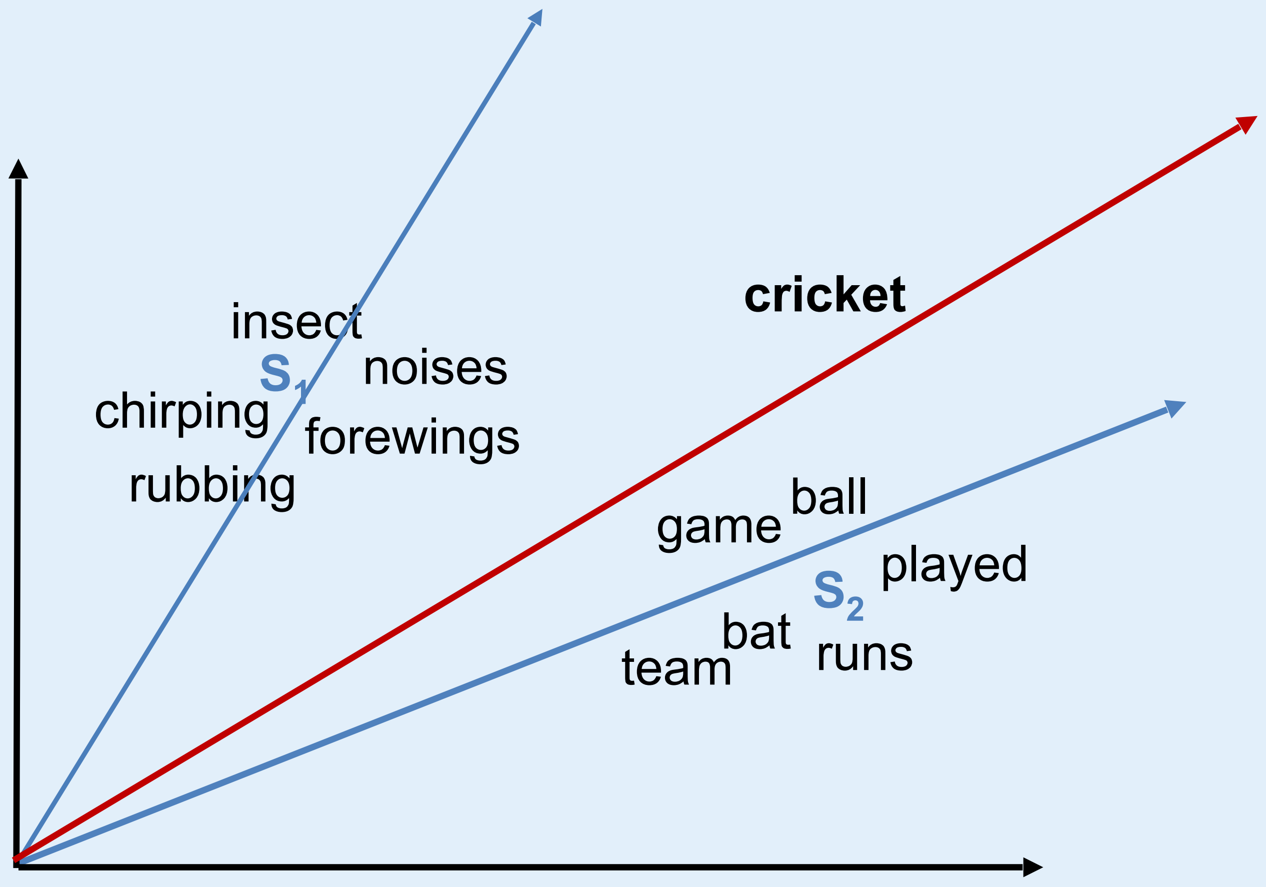
$$MFS_w = \underset{S_i}{\operatorname{argmax}} \cos(vec(W), vec(S_i))$$

  - $vec(W)$ - word embedding of a word W
  - S_i- ith sense of word W
  - $vec(S_i)$ - sense embedding for S_i

MFS Detection


cricket
S_1
S_2
02232196: cricket (leaping insect; male makes chirping noises by rubbing the forewings together)
00477400: cricket (a game played with a ball and bat by two teams of 11 players; teams take turns trying to score runs)
insect
chirping
noises
forewings
rubbing
Gloss SenseBag (S_1)
game
played
ball
runs
team
bat
Gloss SenseBag (S_2)

MFS Detection contd..



Experiments

A. Experiments on WSD

   1. Experiments on WSD using Skip-Gram model
      - Hindi (Newspaper)
      - English (SENSEVAL-2 and SENSEVAL-3)
   2. Experiments on WSD using different word vector models
   3. Comparing WSD results using different sense vector models
      - Retrofitting Sense Vector Model (English)

B. Experiments on selected words (34 polysemous words from SENSEVAL-2 corpus)

   1. Experiments using different word vector models
   2. Comparing results with various sizes of vector dimensions

Experiments

A. Experiments on WSD

   1. **Experiments on WSD using Skip-Gram model**

      - **Hindi (Newspaper)**
      - **English (SENSEVAL-2 and SENSEVAL-3)**

[A.1] Experiments on WSD using skip-gram

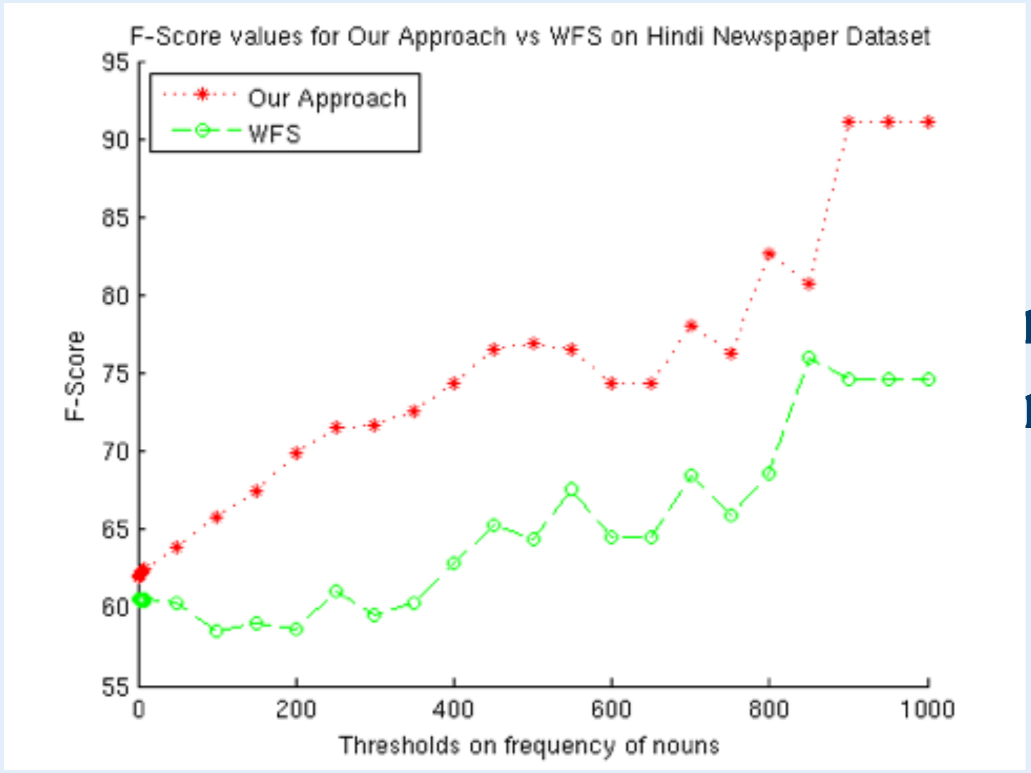
- Training of word embeddings:
  - Hindi: Bojar (2014) corpus (44 M sentences)
  - English: Pre-trained Google-News word embeddings
- Datasets used for WSD:
  - Hindi: Newspaper dataset
  - English: SENSEVAL-2 and SENSEVAL-3
- Experiments are restricted to only polysemous nouns.

[A.1] Results on WSD

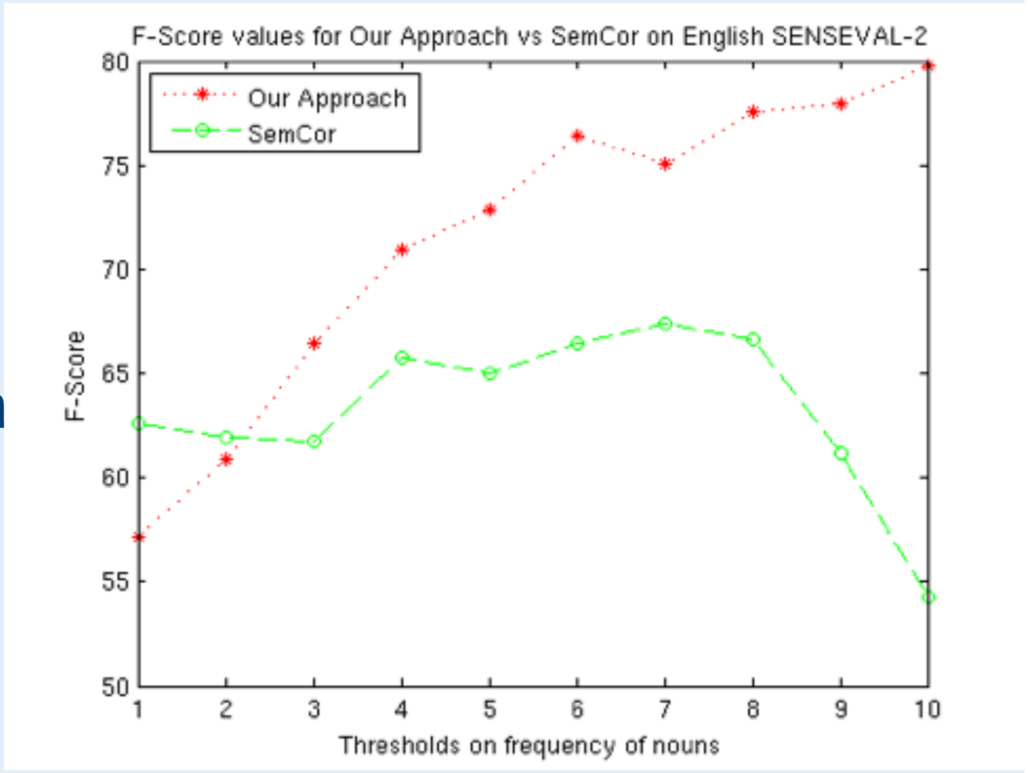
HINDI WSD	Newspaper dataset		
	Precision	Recall	F-Score
UMFS-WE	62.43	61.58	62.00
WFS	61.73	59.31	60.49

ENGLISH WSD	SENSEVAL-2 dataset			SENSEVAL-3 dataset		
	Precision	Recall	F-Score	Precision	Recall	F-Score
UMFS-WE	52.39	52.27	52.34	43.34	43.22	43.28
WFS	61.72	58.16	59.88	66.57	64.89	65.72

[A.1] Results on WSD contd..



r
n



Hindi WSD

English WSD

[A.1] Results on WSD contd..

- WordNet feature selection for sense embeddings creation

SB: Synset Bag

GB: Gloss Bag

EB: Example Bag

PSB: Parent Synset Bag

PGB: Parent Gloss Bag

PEB: Parent Example Bag

Sense Vectors Using WordNet features	**Precision**	**Recall**	**F-measure**
SB	51.73	38.13	43.89
SB+GB	53.31	52.39	52.85
SB+GB+EB	56.61	55.84	56.22
SB+GB+EB+PSB	59.53	58.72	59.12
SB+GB+EB+PGB	60.57	59.75	60.16
SB+GB+EB+PEB	60.12	59.3	59.71
SB+GB+EB+PSB+PGB	57.59	56.81	57.19
SB+GB+EB+PSB+PEB	58.93	58.13	58.52
SB+GB+EB+PGB+PEB	**62.43**	**61.58**	**62**
SB+GB+EB+PSB+PGB+PEB	58.56	57.76	58.16

Table: Hindi WSD results using various WordNet features for Sense Embedding creation

Experiments

A. Experiments on WSD

   1. Experiments on WSD using Skip-Gram model

      - Hindi (Newspaper)
      - English (SENSEVAL-2 and SENSEVAL-3)

   2. **Experiments on WSD using different word vector models**

[A.2] Experiments on WSD using various Word Vector models

- MFS results is compared on various word vector models as listed below:

Word Vector Model	Dimensions
SkipGram-Google-News (Mikolov et. al, 2013)	300
Senna (Collobert et. al, 2011)	50
MetaOptimize (Turian et. al, 2010)	50
RNN (Mikolov et. al, 2011)	640
Glove (Pennington et. al, 2014)	300
Global Context (Huang et. al, 2013)	50
Multilingual (Faruqui et.al, 2014)	512
SkipGram-BNC (Mikolov et. al, 2013)	300
SkipGram-Brown (Mikolov et. al, 2013)	300

[A.2] Experiments on WSD using various Word Vector models contd..

WordVector	Noun	Adj	Adv	Verb
SkipGram-Google-News	54.49	50.56	**47.66**	20.66
Senna	54.49	40.44	28.97	21.9
RNN	39.07	28.65	40.18	19.42
MetaOptimize	33.73	36.51	32.71	19.83
Glove	**54.69**	**49.43**	39.25	18.18
Global Context	48.3	32.02	31.77	20.66
SkipGram-BNC	53.03	48.87	39.25	**23.14**
SkipGram-Brown	30.29	48.87	27.10	13.29

Table: English WSD results for words with corpus frequency > 2

Experiments

A. Experiments on WSD

1. Experiments on WSD using Skip-Gram model
   - Hindi (Newspaper)
   - English (SENSEVAL-2 and SENSEVAL-3)
2. Experiments on WSD using different word vector models
3. **Comparing WSD results using different sense vector models**
   - **Retrofitting Sense Vector Model (English)**

[A.3] Results on WSD

WordVector	SenseVector	Noun	Adj	Adv	Verb
SkipGram-Google-News	Our model	**58.87**	53.53	**46.34**	20.49
	Retrofitting	47.84	**57.57**	32.92	**21.73**
Senna	Our model	**61.29**	43.43	**21.95**	**24.22**
	Retrofitting	6.9	**68.68**	**21.95**	1.86
RNN	Our model	**42.2**	26.26	**40.24**	**21.11**
	Retrofitting	10.48	**62.62**	21.95	1.24
MetaOptimize	Our model	**37.9**	50.5	**31.7**	**18.01**
	Retrofitting	10.48	**62.62**	21.95	1.24
Glove	Our model	**58.33**	53.33	**39.02**	**17.39**
	Retrofitting	9.94	**62.62**	21.95	1.24
Global Context	Our model	53.22	37.37	**24.39**	19.25
	Retrofitting	12.36	**68.68**	21.95	1.24
SkipGram-Brown	Our model	29.31	60.6	**23.17**	11.42
	Retrofitting	11.49	**68.68**	21.95	1.26

Table: English WSD results for words with corpus frequency > 2

Experiments

A. Experiments on WSD
   1. Experiments on WSD using Skip-Gram model
      - Hindi (Newspaper)
      - English (SENSEVAL-2 and SENSEVAL-3)
   2. Experiments on WSD using different word vector models
   3. Comparing WSD results using different sense vector models
      - Retrofitting Sense Vector Model (English)

B. **Experiments on selected words (34 polysemous words from SENSEVAL-2 corpus)**
   1. **Experiments using different word vector models**

[B.1] Experiments on selected words

- 34 polysemous nouns, where each one has atleast two senses and which have occurred at least twice in the SENSEVAL-2 dataset are chosen

Token	Senses	Token	Senses
church	4	individual	2
field	13	child	4
bell	10	risk	4
rope	2	eye	5
band	12	research	2
ringer	4	team	2
tower	3	version	6
group	3	copy	3
year	4	loss	8
vicar	3	colon	5
sort	4	leader	2
country	5	discovery	4
woman	4	education	6
cancer	5	performance	5
cell	7	school	7
type	6	pupil	3
growth	6	student	2

[B.1] MFS Results on selected words

Word Vectors	Accuracy
SkipGram-BNC	63.63
SkipGram-Brown	48.38
SkipGram-Google-News	60.6
Senna	57.57
Glove	**66.66**
Global Context	51.51
Metaoptimize	27.27
RNN	51.51
Multilingual	63.4

Table: English WSD results for selected words from SENSEVAL-2 dataset

Experiments

A. Experiments on WSD
   1. Experiments on WSD using Skip-Gram model
      - Hindi (Newspaper)
      - English (SENSEVAL-2 and SENSEVAL-3)
   2. Experiments on WSD using different word vector models
   3. Comparing WSD results using different sense vector models
      - Retrofitting Sense Vector Model (English)

B. Experiments on selected words (34 polysemous words from SENSEVAL-2 corpus)
   1. Experiments using different word vector models
   2. **Comparing results with various sizes of vector dimensions**

[B.2] Comparing MFS results with various sizes of vector dimensions

Word Vectors	Accuracy
SkipGram-BNC-1500	60.61
SkipGram-BNC-1000	60.61
SkipGram-BNC-500	66.67
SkipGram-BNC-400	**69.69**
SkipGram-BNC-300	63.64
SkipGram-BNC-200	60.61
SkipGram-BNC-100	48.49
SkipGram-BNC-50	51.52

Summary

- An unsupervised approach is designed for finding the MFS by using word embeddings.
- Tested MFS results on WSD and some selected words.
- Performance is compared with different word vector models and various size of the dimensions.
- Our sense vector model always show better results as on nouns, adjectives and adverbs as compared to *retrofitting* model.
- Approach can be easily ported to various domains and across languages.

References

- Buitelaar, Paul, and Bogdan Sacaleanu. 2001. “Ranking and selecting synsets by domain relevance”. Proceedings of WordNet and Other Lexical Resources: Applications, Extensions and Customizations, NAACL 2001 Workshop.
- Mohammad, Saif, and Graeme Hirst. 2006. “Determining Word Sense Dominance Using a Thesaurus”. EACL.
- Lapata, Mirella, and Chris Brew. 2004. “Verb class disambiguation using informative priors”. Computational Linguistics 30.1 (2004): 45-73.
- O. Bojar, V. Diatka, P. Rychlý, P. Stranák, V. Suchomel, A. Tamchyna, and D. Zeman. 2014. “HindEnCorp- Hindi-English and Hindi-only Corpus for Machine Translation”. In Proceedings of LREC. 2014, 3550-3555.
- Diana Mccarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. “Finding predominant word senses in untagged text”. In In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, pages 280–287.
- Xinxiong Chen, Zhiyuan Liu and Maosong Sun. 2014. “A Unified Model for Word Sense Representation and Disambiguation”, Proceedings of ACL 2014.
- Tang, D.,Wei, F., Yang, N., Zhou, M., Liu, T., and Qin, B. 2014. “Learning sentiment-specific word embedding for twitter sentiment classification”. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1555–1565.

References contd..

- Manaal Faruqui and Chris Dyer. 2014. “Community Evaluation and Exchange of Word Vector”,s at wordvectors.org ,Proceedings of System Demonstrations, ACL 2014
- Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Honza Cernocky. 2011. “RNNLM - Recurrent Neural Network Language Modeling Toolkit”, In: ASRU 2011
- Jauhar, Sujay Kumar, Chris Dyer, and Eduard Hovy. 2015. "Ontologically Grounded Multi-sense Representation Learning for Semantic Vector Space Models.”, ACL 2015.

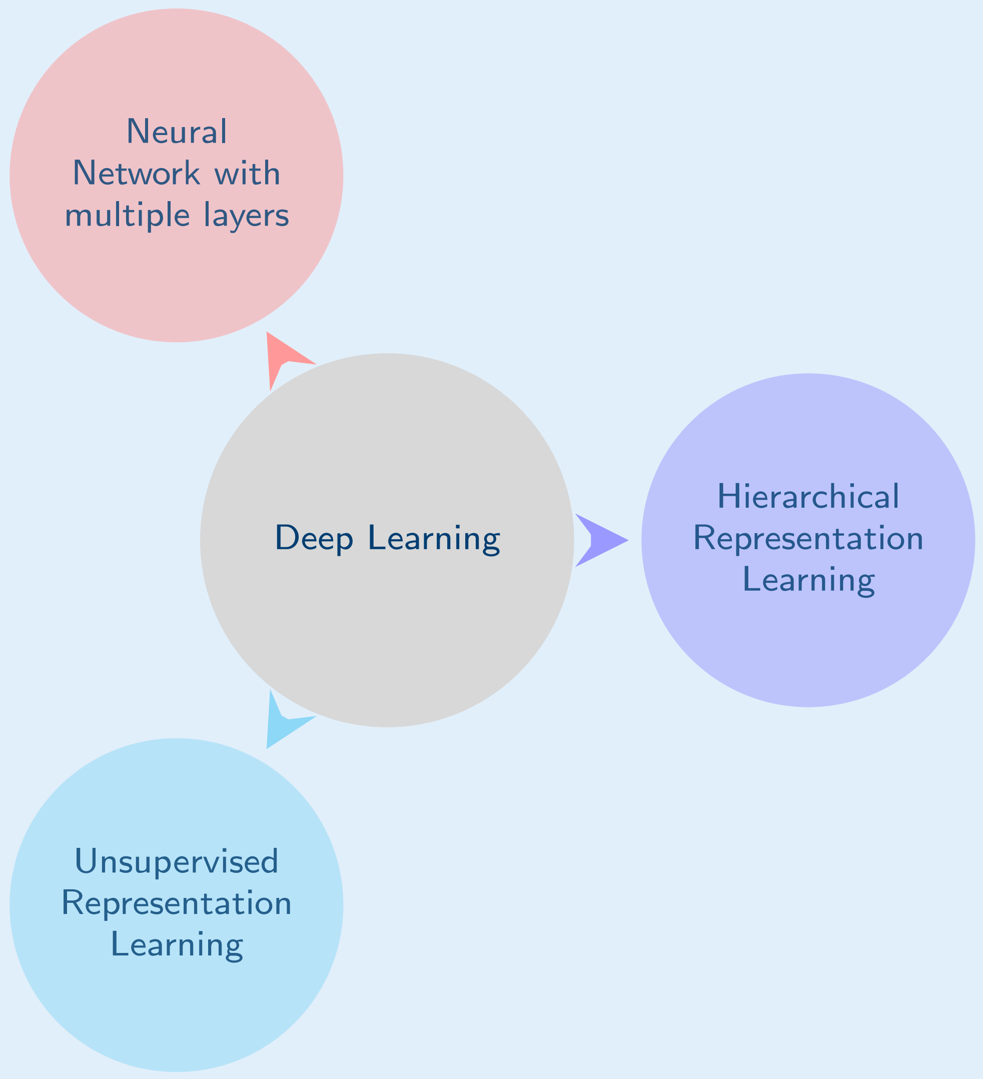
Q & A

What is Deep Learning?

Wikipedia

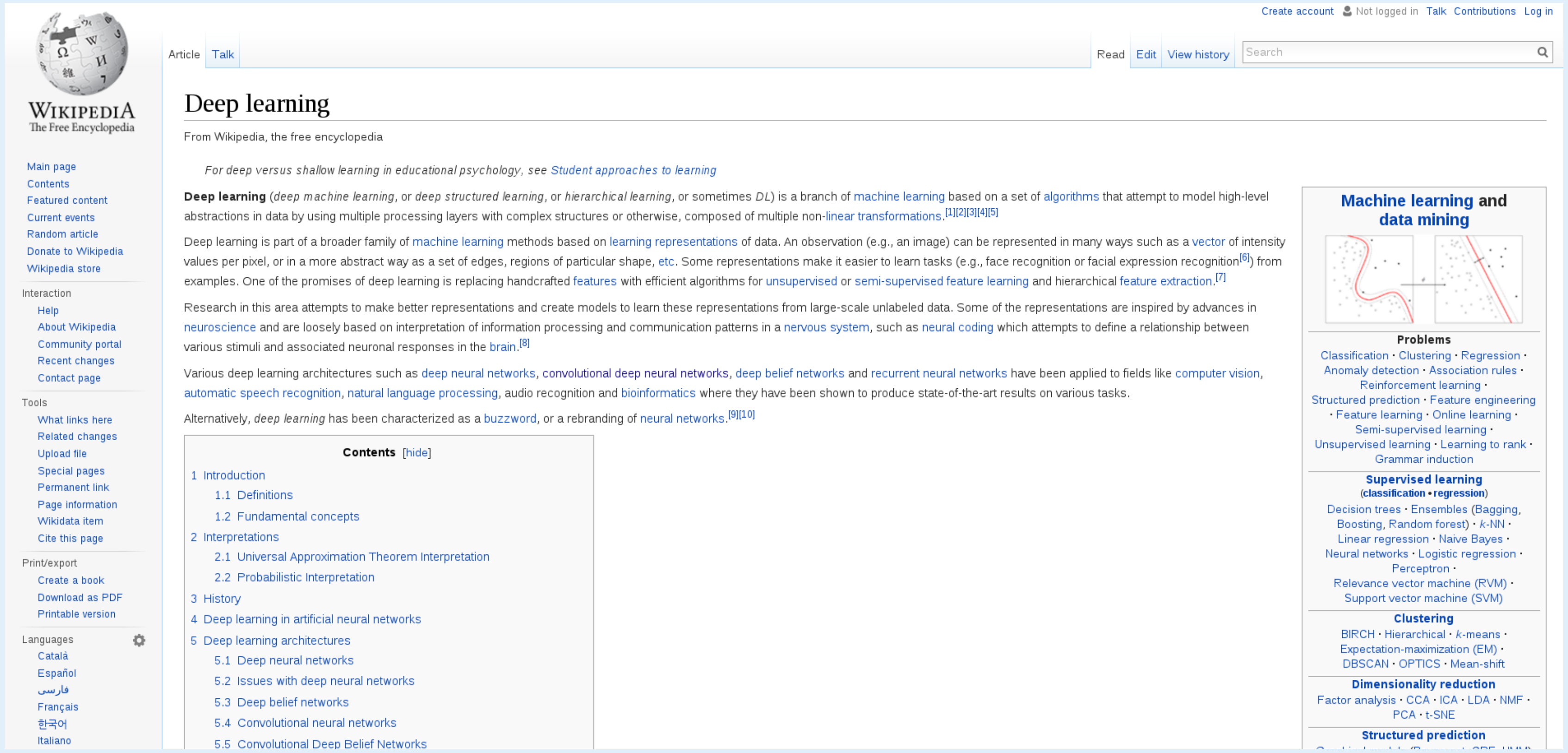
Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers with complex structures or otherwise, composed of multiple non-linear transformations

What is Deep Learning?


Neural Network with multiple layers
Deep Learning
Hierarchical Representation Learning
Unsupervised Representation Learning

Why Deep Learning?

Lots of Unlabeled data



Create account · Not logged in · Talk · Contributions · Log in

Article · Talk · Read · Edit · View history · Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Català
Español
فارسی
Français
한국어
Italiano

Deep learning

From Wikipedia, the free encyclopedia

For deep versus shallow learning in educational psychology, see Student approaches to learning

Deep learning (*deep machine learning*, or *deep structured learning*, or *hierarchical learning*, or sometimes *DL*) is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers with complex structures or otherwise, composed of multiple non-linear transformations.[1][2][3][4][5]

Deep learning is part of a broader family of machine learning methods based on learning representations of data. An observation (e.g., an image) can be represented in many ways such as a vector of intensity values per pixel, or in a more abstract way as a set of edges, regions of particular shape, etc. Some representations make it easier to learn tasks (e.g., face recognition or facial expression recognition[6]) from examples. One of the promises of deep learning is replacing handcrafted features with efficient algorithms for unsupervised or semi-supervised feature learning and hierarchical feature extraction.[7]

Research in this area attempts to make better representations and create models to learn these representations from large-scale unlabeled data. Some of the representations are inspired by advances in neuroscience and are loosely based on interpretation of information processing and communication patterns in a nervous system, such as neural coding which attempts to define a relationship between various stimuli and associated neuronal responses in the brain.[8]

Various deep learning architectures such as deep neural networks, convolutional deep neural networks, deep belief networks and recurrent neural networks have been applied to fields like computer vision, automatic speech recognition, natural language processing, audio recognition and bioinformatics where they have been shown to produce state-of-the-art results on various tasks.

Alternatively, *deep learning* has been characterized as a buzzword, or a rebranding of neural networks.[9][10]

Contents [hide]

Machine learning and data mining

Problems
Classification · Clustering · Regression · Anomaly detection · Association rules · Reinforcement learning · Structured prediction · Feature engineering · Feature learning · Online learning · Semi-supervised learning · Unsupervised learning · Learning to rank · Grammar induction

Supervised learning
(**classification** • **regression**)
Decision trees · Ensembles (Bagging, Boosting, Random forest) · *k*-NN · Linear regression · Naive Bayes · Neural networks · Logistic regression · Perceptron · Relevance vector machine (RVM) · Support vector machine (SVM)

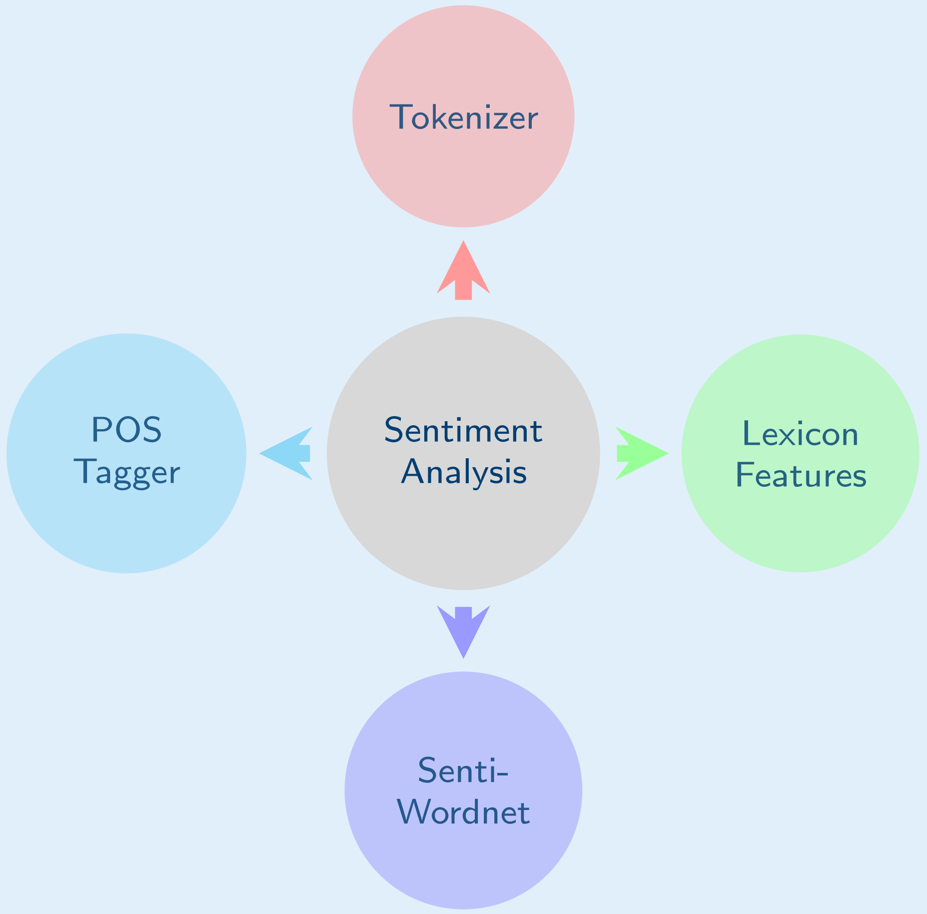
Clustering
BIRCH · Hierarchical · *k*-means · Expectation-maximization (EM) · DBSCAN · OPTICS · Mean-shift

Dimensionality reduction
Factor analysis · CCA · ICA · LDA · NMF · PCA · t-SNE

Structured prediction

Why Deep Learning?

Success of Traditional Approach depends on Handcrafted Features and Knowledge Resources [1]



[1]Sentiment Symposium Tutorial: http://sentiment.christopherpotts.net/

Why Deep Learning?

Many tasks are inherently complex leading to hierarchical way of solving

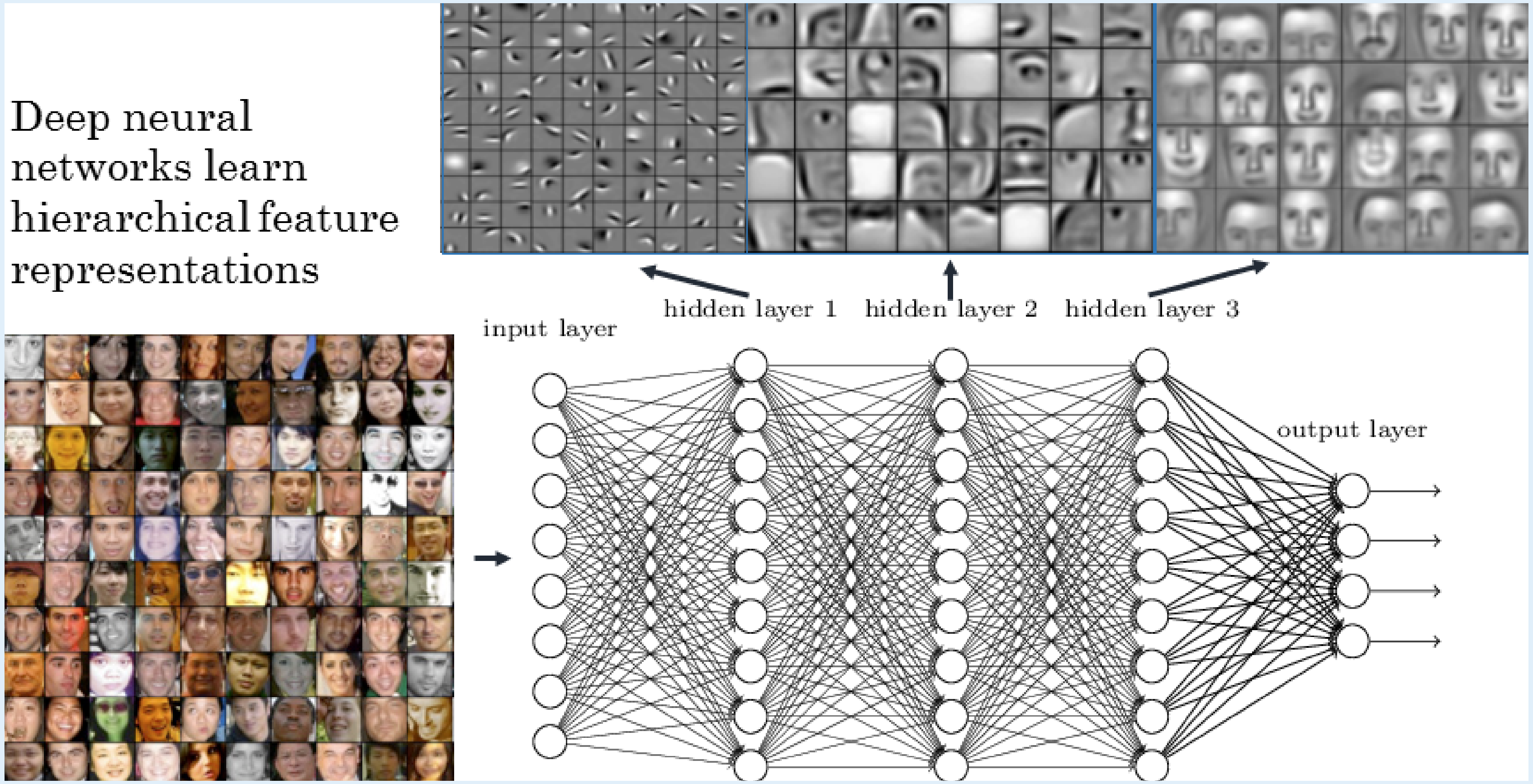


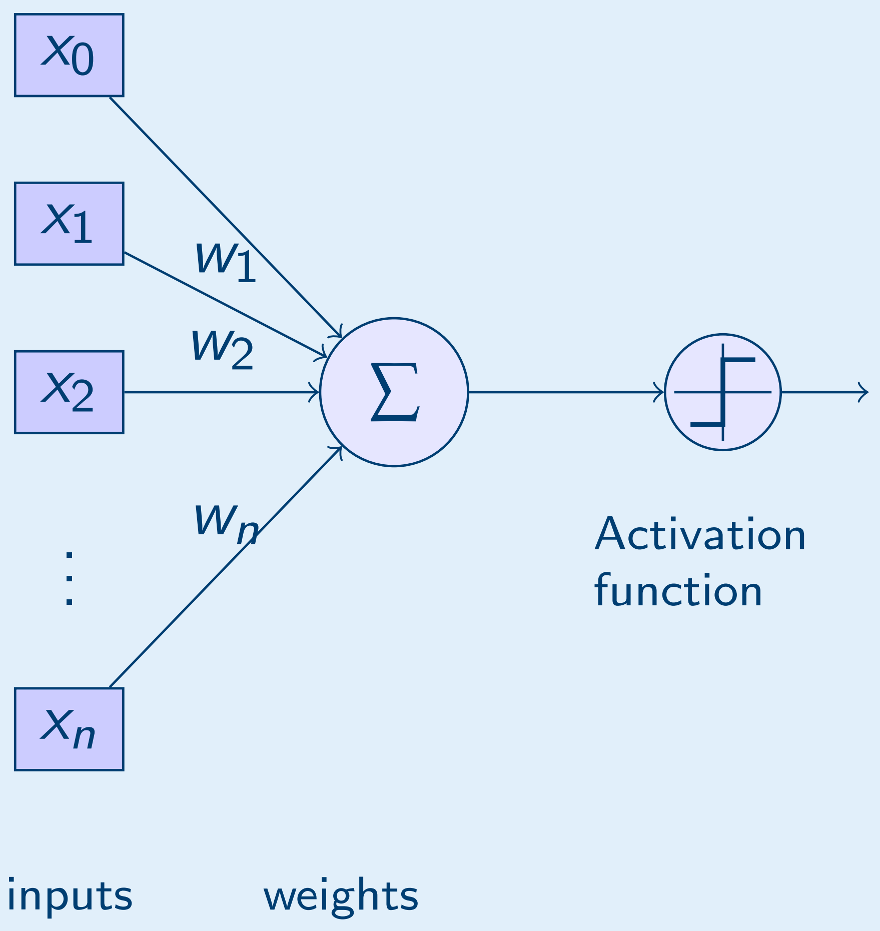
Figure: Hierachical way of learning features for Image Classification [Lee et al. n.d.]

Plan

Perceptron [Rosenblatt 1962]

- Given a set of input/label pairs $(x^1, y^1), \ldots, (x^n, y^n)$
- Learn a function to classify the problem
- Learn a set of weights $(w_1, \ldots, w_m)$ for the input feature
- $$f(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{m} w_i x_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Perceptron



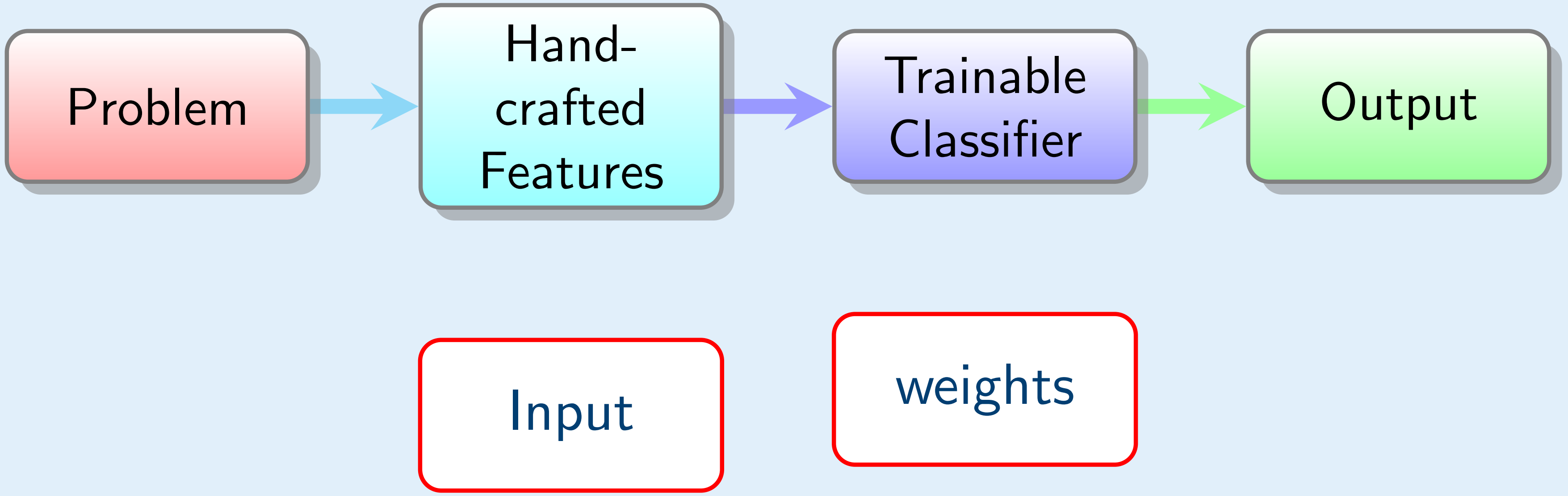
x_0
x_1
x_2
$\vdots$
x_n
w_1
w_2
w_n
Σ
Activation function
inputs
weights

Training a Perceptron

Algorithm 1 Perceptron Training Algorithm

```
$w \leftarrow zeros()$                                  ▷ Initialize the feature weights to zero
**for** every example $(x, y) \in$ Dataset $D$ **do**
    $t \leftarrow f(\sum_{i=1}^{m} w_i x_i)$            ▷ Calculate Prediction
    $w = w + \alpha(y - t)x$                            ▷ Update Feature Weights
```

Perceptron Algorithm



Perceptron

- The perceptron calculates, $y = \sum_{i=1}^{m} w_i x_i + b$
- This is similar to $y = mx + c$ which is equation of a line in 2d and hyperplane in general
- Divide the input feature space into two regions, (positive and negative class regions)

Disadvantages of Perceptron Algorithm

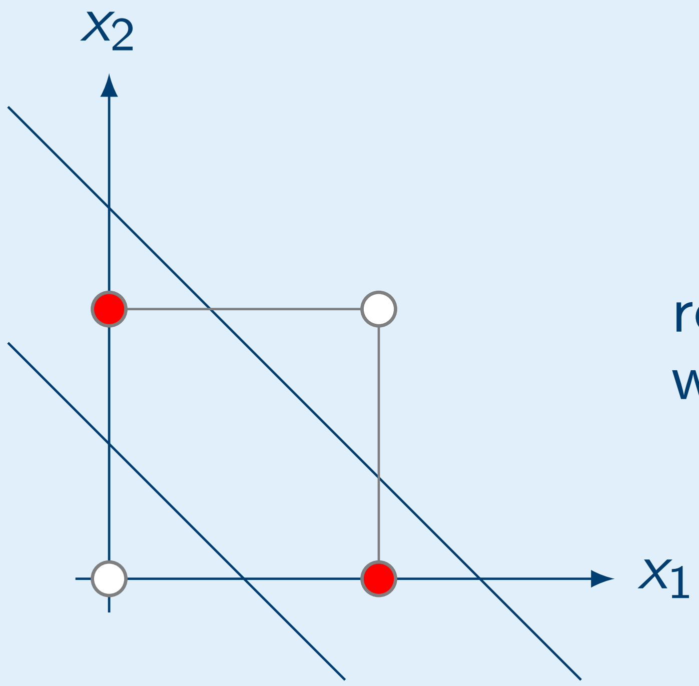
- Cannot learn non-linear functions
- Famous XOR example

Disadvantages of Perceptron Algorithm

Input		Output
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

- Train a perceptron to replicate XOR gate
- Learn weights w_1, w_2

Disadvantages of Perceptron Algorithm

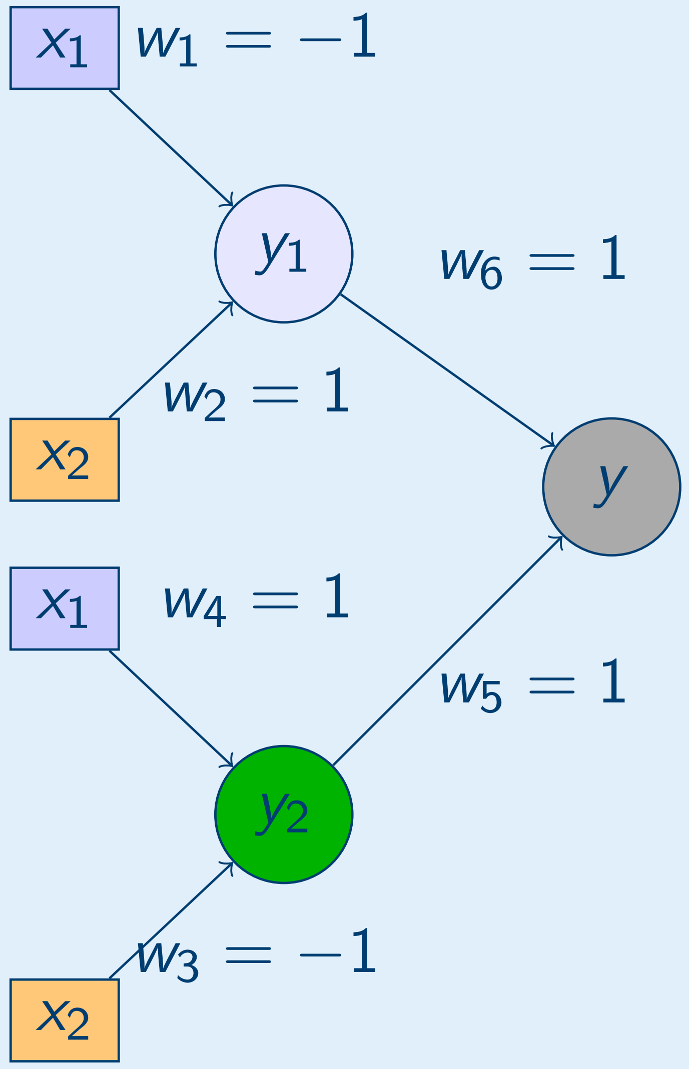


red circles indicate 1
white circles indicate 0

Disadvantages of Perceptron Algorithm

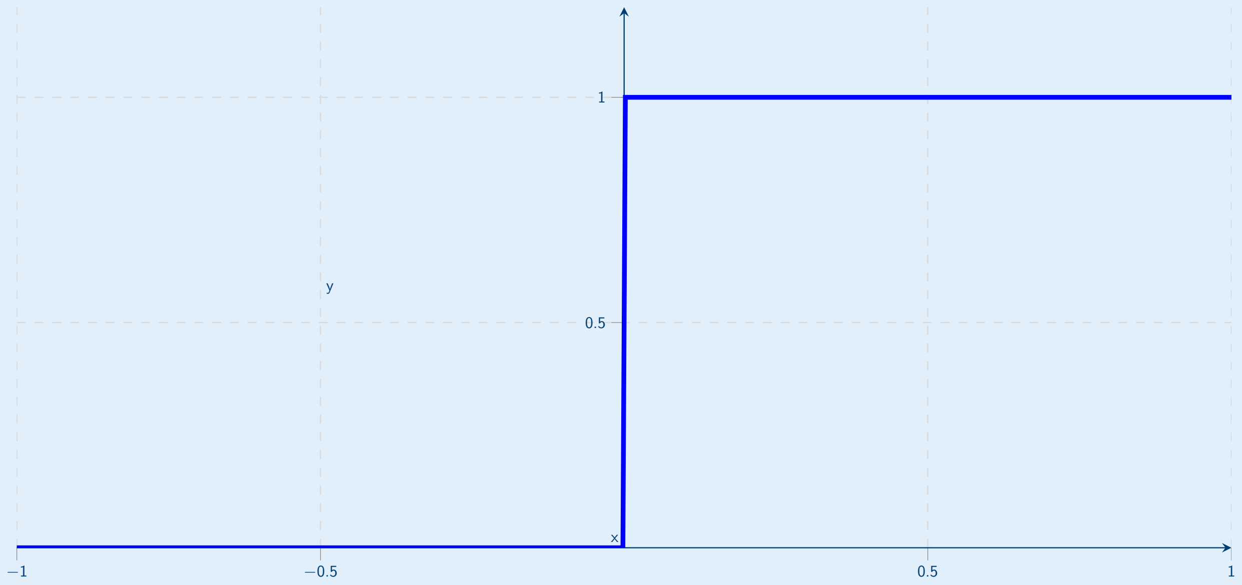
- A single perceptron cannot learn an XOR function
- Need two decision boundaries
- Need multiple perceptrons
- What about hierarchy of connected perceptrons?

Multiple Perceptrons for XOR Problem



Feedforward Neural Network

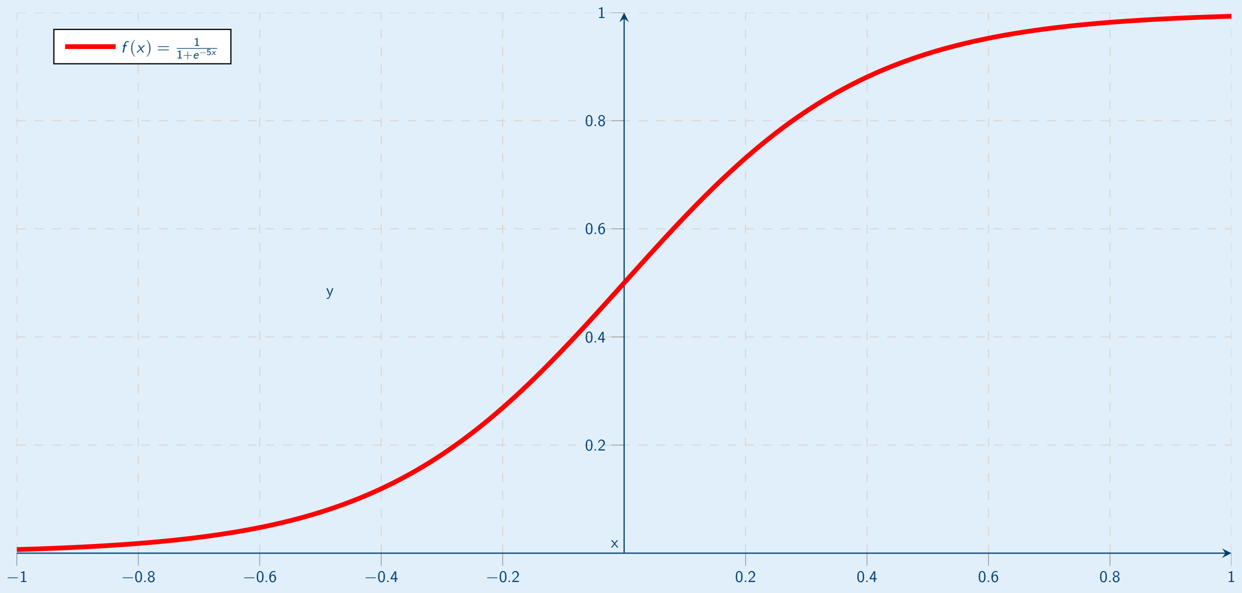
- The previous model used Threshold function as activation function



- We need a smooth approximation to this function

Feedforward Neural Network

- One such approximation is Sigmoidal units

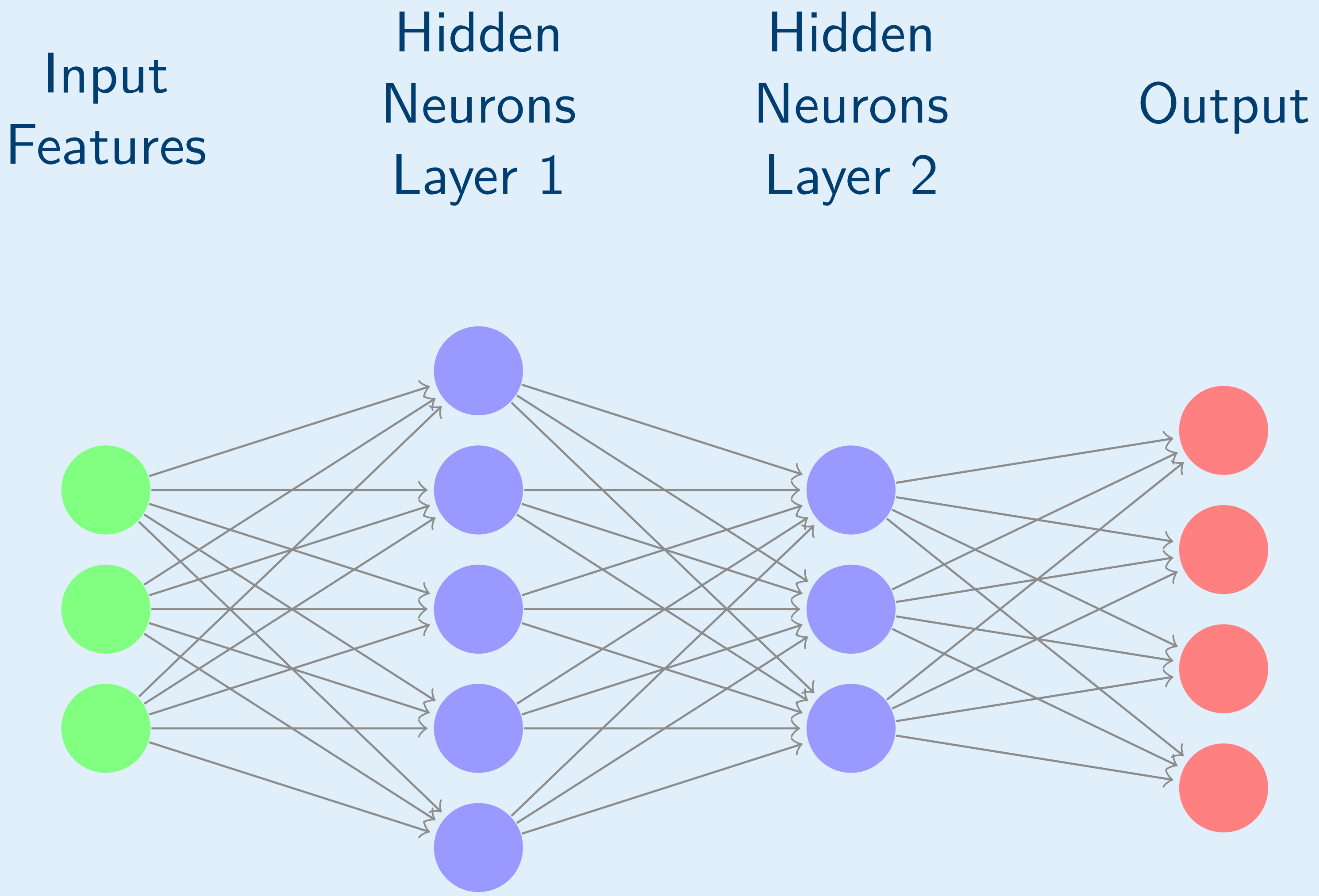


- Any non-linear activations like tanh, HardTanh, RELUs etc. are used

Feedforward Neural Network

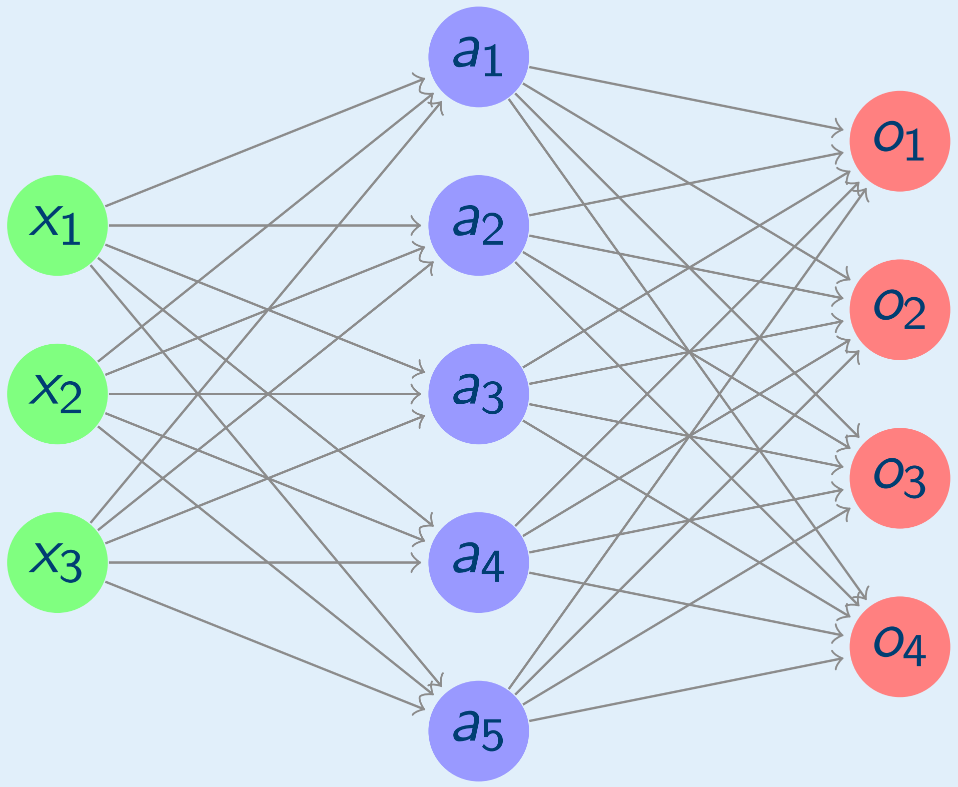
- Many layers of perceptrons can be connected in a sequential way
- Each layer can have multiple perceptrons taking the same input
- Can be used to solve complex tasks
- The architecture gives rise to FeedForward Neural Network (FFNN)

Feedforward Neural Network



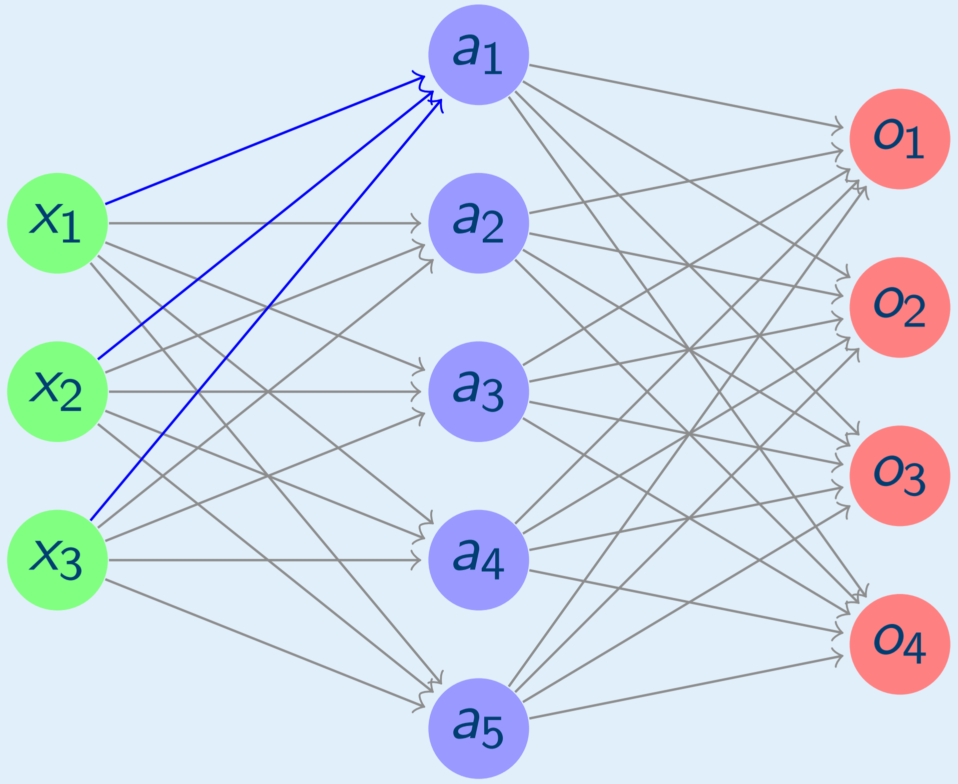
Feedforward Neural Network: Forward Propagation

- Consider a simple FeedForward Neural Network with one hidden layer
- f is the hidden layer non-linear activation function
- g is the output non-linear activation function



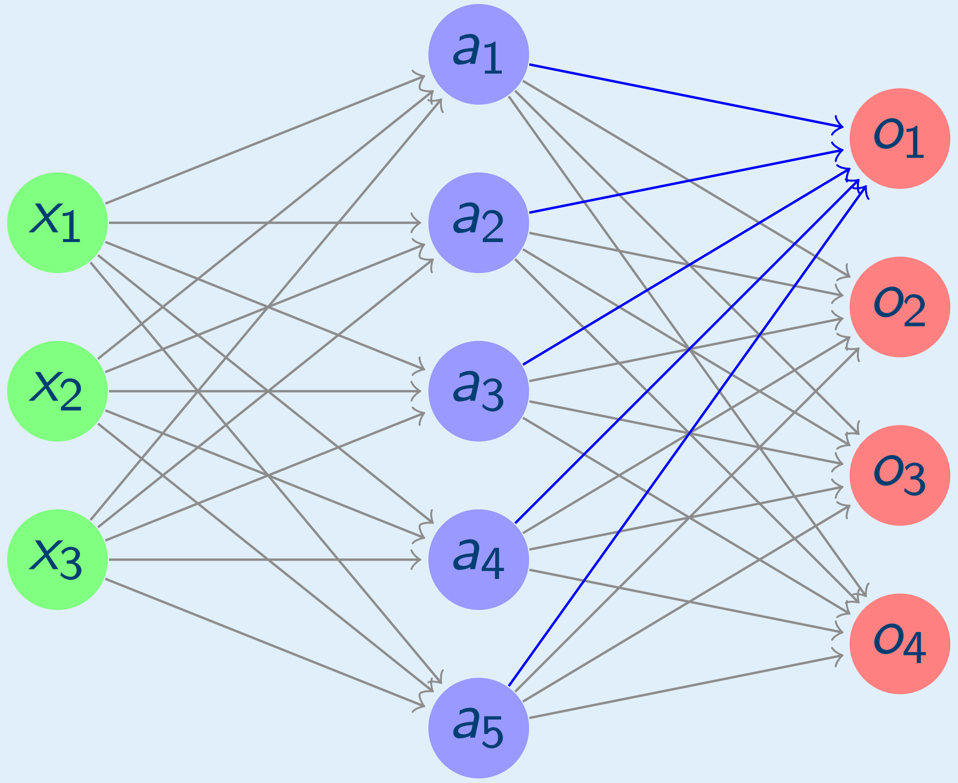
Feedforward Neural Network: Forward Propagation

- Let $X = (x_1, \ldots, x_n)$ be the set of input features
- hidden layer activation neurons, $a_j = f\left(\sum_{i=1}^{n} W_{ji} x_i\right), \ \forall j \in 1, \ldots h$



Feedforward Neural Network: Forward Propagation

- Let $a = (a_1, \ldots, a_h)$ be the set of hidden layer features
- output neurons, $o_k = g(\sum_{j=1}^{h} U_{kj} a_j), \ \forall k \in 1, \ldots K$



FFNN: Backpropagation Algorithm [Rumelhart, Geoffrey E. Hinton, and Williams 1986]

- Adjust weights W and U to minimize the error on training set
- Define the error to be squared loss between predictions and true output

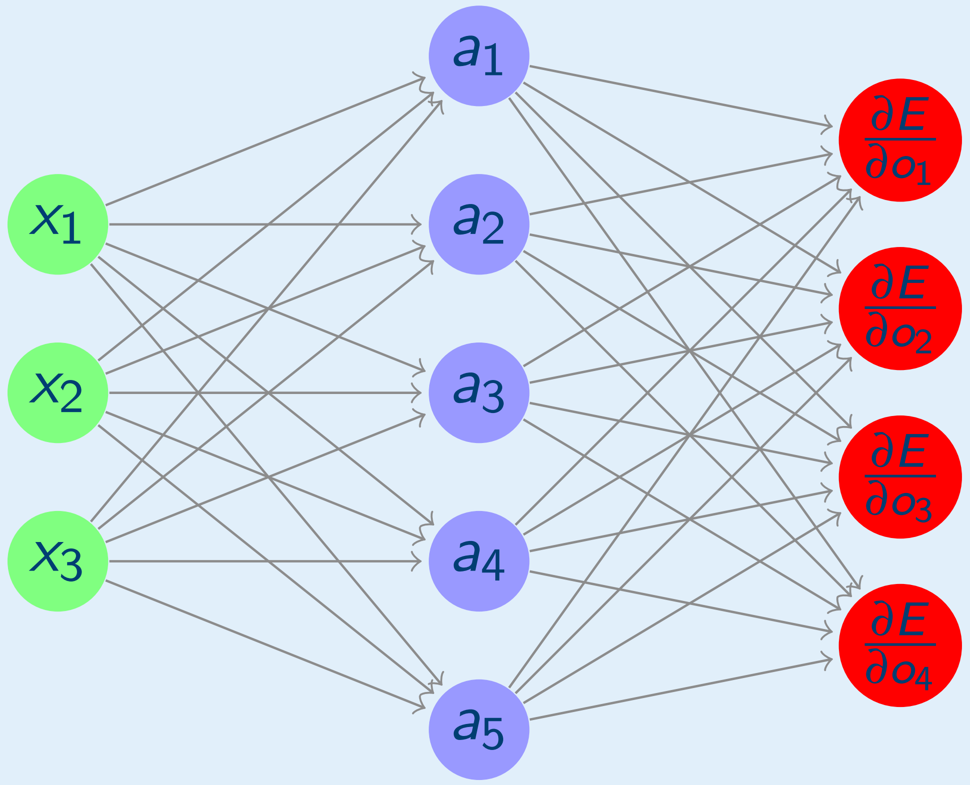
$$E = \frac{1}{2}\text{Error}^2 = \frac{1}{2}(y - o)^2 \tag{1}$$

- Gradient w.r.t to output is,

$$\frac{\partial E}{\partial o_k} = \frac{1}{2} \times 2 \times (y_k - o_k) = (y_k - o_k) \tag{2}$$

FFNN: Backpropagation Algorithm

- We have the errors calculated at output neurons



x_1
x_2
x_3
a_1
a_2
a_3
a_4
a_5
$\frac{\partial E}{\partial o_1}$
$\frac{\partial E}{\partial o_2}$
$\frac{\partial E}{\partial o_3}$
$\frac{\partial E}{\partial o_4}$

- Send the error to lower layers

FFNN: Backpropagation Algorithm

- Calculate gradient w.r.t to parameters U

$$\frac{\partial E}{\partial o_k} = \frac{1}{2} \times 2 \times (y_k - o_k) = (y_k - o_k)$$

- $o_k = g(\sum_{j=1}^{h} U_{kj} a_j), \ \forall k \in 1, \dots K$

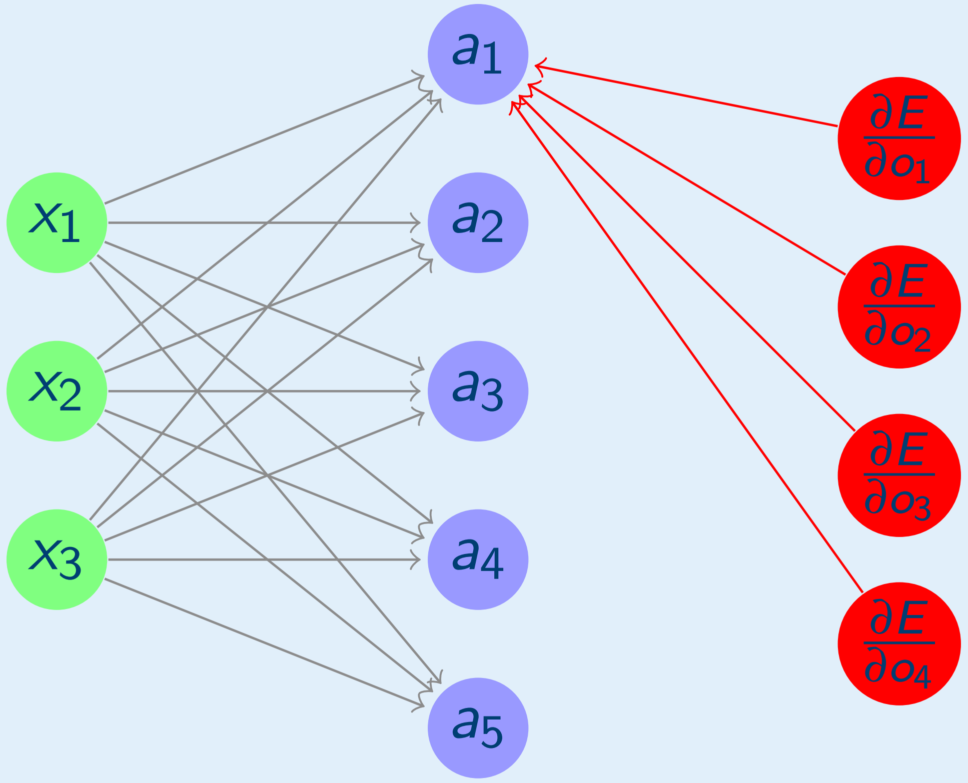
$$\frac{\partial E}{\partial U_{kj}} = \frac{\partial E}{\partial o_k} \times g'(\sum_{j=1}^{h} U_{kj} a_j) \times a_j \tag{3}$$

- Update for U_{kj} will be,

$$U_{kj} = U_{kj} - \eta \times \frac{\partial E}{\partial U_{kj}} \tag{4}$$

FFNN: Backpropagation of errors

- Updation of parameters indicated by red lines


x_1
x_2
x_3
a_1
a_2
a_3
a_4
a_5
$\frac{\partial E}{\partial o_1}$
$\frac{\partial E}{\partial o_2}$
$\frac{\partial E}{\partial o_3}$
$\frac{\partial E}{\partial o_4}$

FFNN: Backpropagation Algorithm

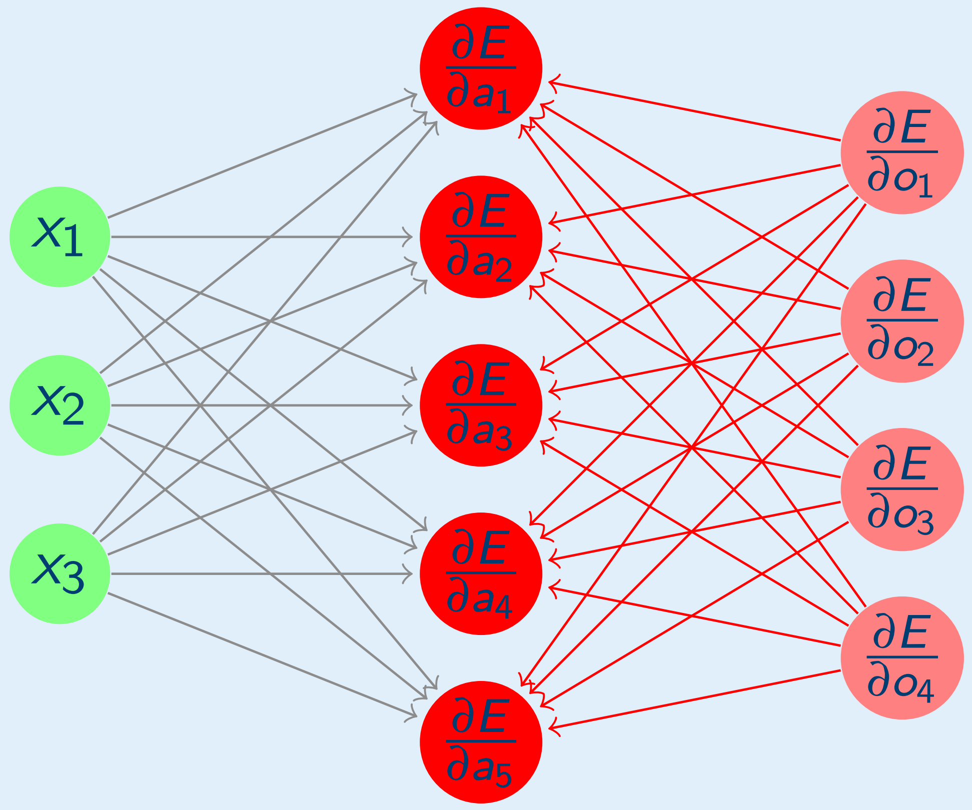
- How to update the parameters W?
- $a_j = f(\sum_{i=1}^{n} W_{ji} x_i)$
- $o_k = g(\sum_{j=1}^{h} U_{kj} a_j)$
- Replacing for a_j, $o_k = g(\sum_{j=1}^{h} U_{kj}\ f(\sum_{i=1}^{n} W_{ji} x_i))$
- Calculate gradient w.r.t a_j

FFNN: Backpropagation Algorithm

- $o_k = g(\sum_{j=1}^{h} U_{kj} a_j)$
- We have calculated $\frac{\partial E}{\partial o_k}$
-
$$\frac{\partial E}{\partial a_j} = \sum_{k=1}^{K} \frac{\partial E}{\partial o_k} \times g' \times U_{kj} \tag{5}$$

FFNN: Backpropagation of errors

- Errors are now accumulated at hidden layer neurons


x_1
x_2
x_3
$\frac{\partial E}{\partial a_1}$
$\frac{\partial E}{\partial a_2}$
$\frac{\partial E}{\partial a_3}$
$\frac{\partial E}{\partial a_4}$
$\frac{\partial E}{\partial a_5}$
$\frac{\partial E}{\partial o_1}$
$\frac{\partial E}{\partial o_2}$
$\frac{\partial E}{\partial o_3}$
$\frac{\partial E}{\partial o_4}$

FFNN: Backpropagation of errors

- We have calculated errors accumulated at each hidden neuron, $\frac{\partial E}{\partial a_j}$
- Use this to update the parametrs W
- $a_j = f(\sum_{i=1}^{n} W_{ji} x_i), \ \forall j \in 1, \dots h$

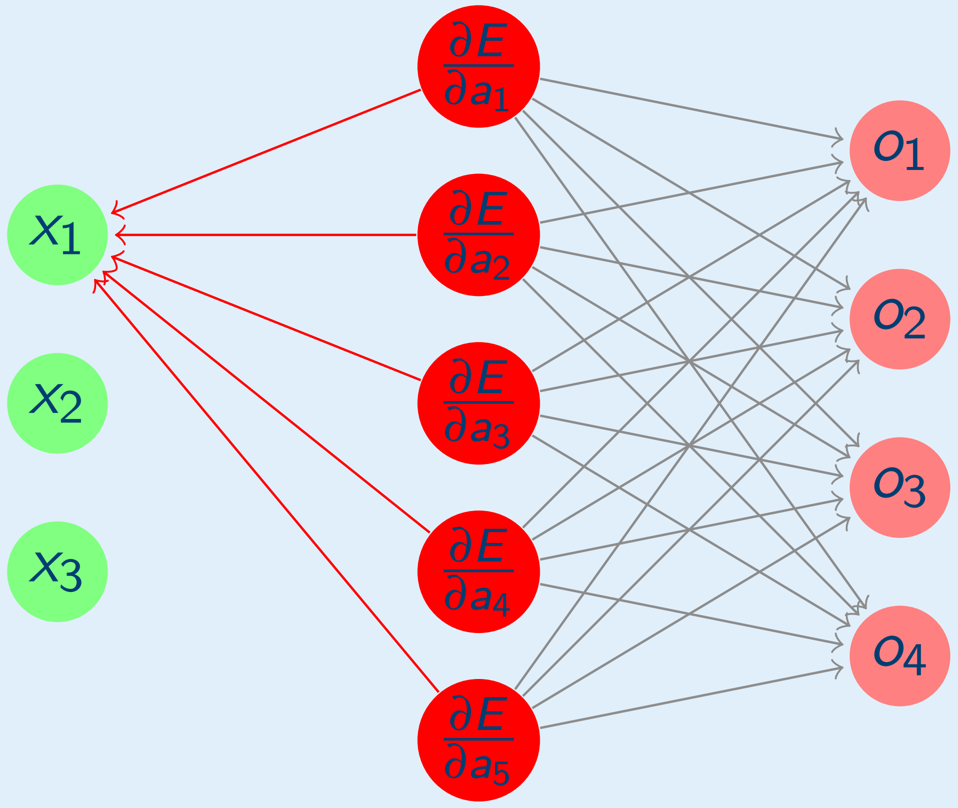
$$\frac{\partial E}{\partial W_{ji}} = \frac{\partial E}{\partial o_j} \times f'(\sum_{i=1}^{n} W_{ji} x_i) \times x_i \tag{6}$$

- Update for W_{ji} will be,

$$W_{ji} = W_{ji} - \eta \times \frac{\partial E}{\partial W_{ji}} \tag{7}$$

FFNN: Backpropagation of errors

- Updation of parameters indicated by red lines

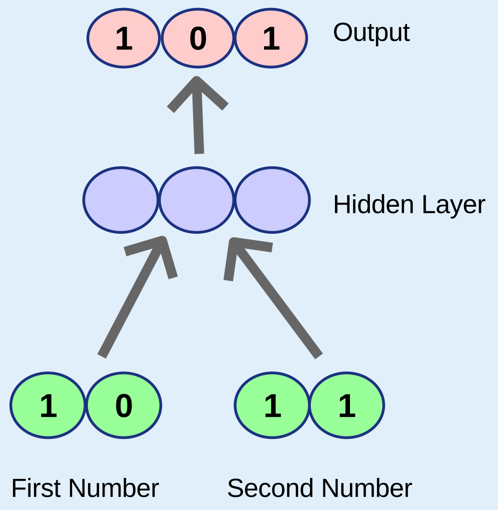

x_1
x_2
x_3
$\frac{\partial E}{\partial a_1}$
$\frac{\partial E}{\partial a_2}$
$\frac{\partial E}{\partial a_3}$
$\frac{\partial E}{\partial a_4}$
$\frac{\partial E}{\partial a_5}$
o_1
o_2
o_3
o_4

FeedForward Neural Network

- The training proceeds in an online fashion i.e, update parameters after every example
- Minibatches are also used (i.e, parameters are updated after seeing k examples)
- Monitor the error on validation set after one complete sweep of training set
- The training repeats until the error on validation set stops to decrease

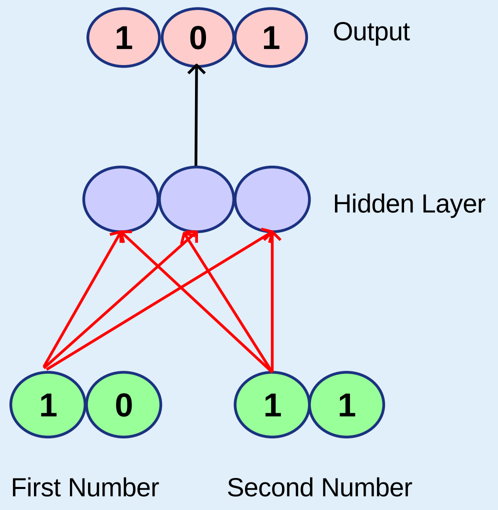
2-bit adder: FeedForward Neural Network

- Train a feedforward network to add two 2-digit binary numbers
- The length of binary numbers needs to be decided



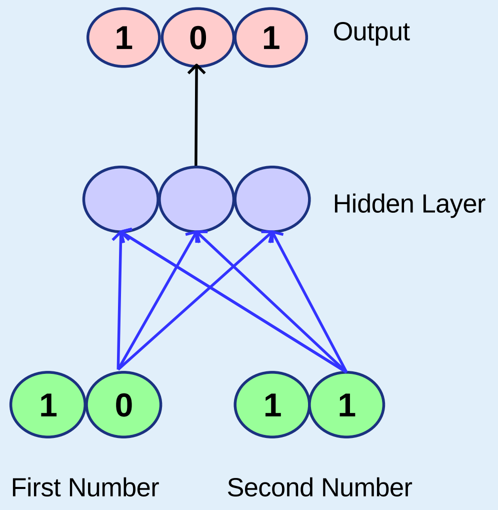
2-bit adder: FeedForward Neural Network

- Different weights are used for different positions of number
- Red connections indicate first digit connections
- Blue connections indicate last digit connections
- red connections vs blue connections
- Ideally we want the same weights to be used for adding bits irrespective of their position



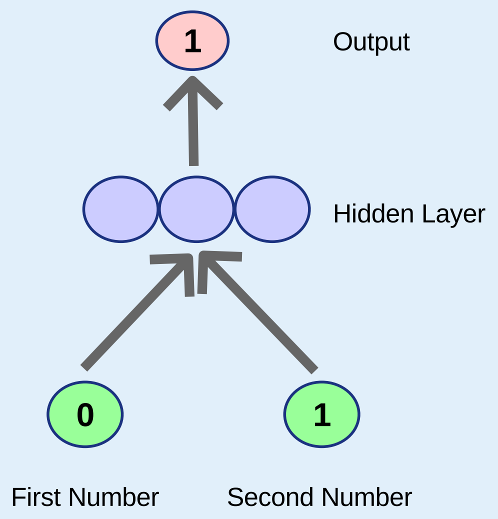
2-bit adder: FeedForward Neural Network

- Different weights are used for different positions of number
- Red connections indicate first digit connections
- Blue connections indicate last digit connections
- red connections vs blue connections
- Ideally we want the same weights to be used for adding bits irrespective of their position



2-bit Adder: FeedForward Neural Network

- The network reads sequentially one bit from two numbers
- The output is the sum of the two digits
- What about *carry* generated from previous addition?
- Can we make the network remember previous carry?

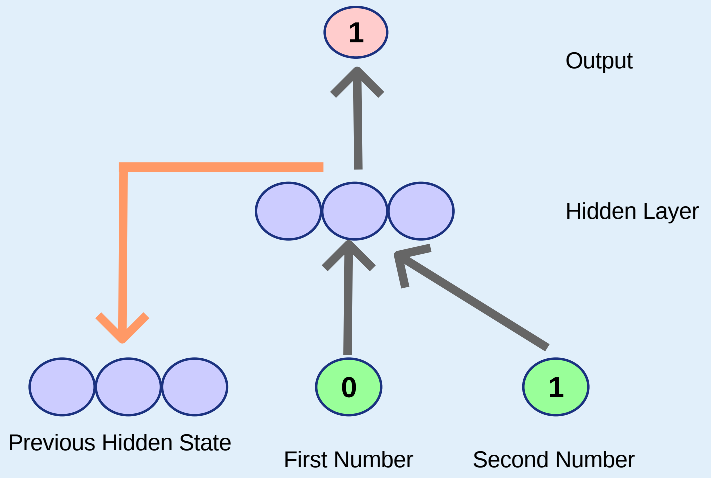


2-bit Adder: Adding Memory

- Make hidden layer neurons have memory
- The hidden layer neurons look at input as well as previous hidden layer state

$$a_j^t = \sum_{i=1}^{2} W_{ji} x_i + \sum_{i=1}^{3} H_{ji} a_i^{t-1}$$

- The hidden layer acts as a state variable
- Depending on the state the predicted output varies for the same set of inputs



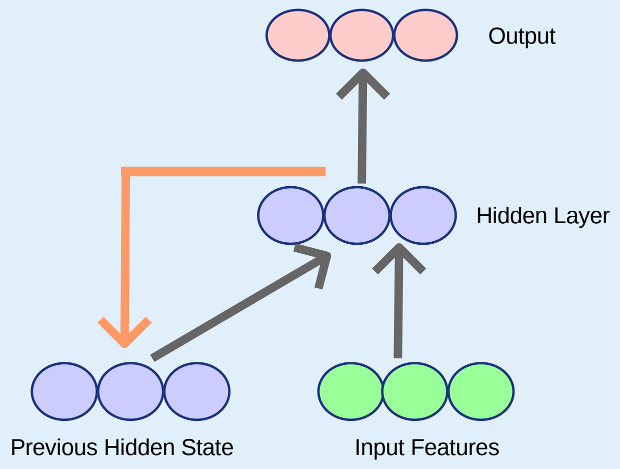
Recurrent Neural Network [Elman 1990]

- Let $x^t = x_1^t, \ldots, x_n^t$ be the input to the network at time t
- The hidden state at current time t is

$$a_j^t = f(\sum_{i=1}^{n} W_{ji} x_i^t + \sum_{i=1}^{h} H_{ji} a_i^{t-1})$$

- This hidden state is then fed to the output layer

$$o_k^t = g(\sum_{j=1}^{h} U_{kj} a_j^t)$$



Recurrent Neural Network: Forward Propagation

Algorithm 2 RNN: Forward Propagation

```
$a_0 \leftarrow zeros()$                                  ▷ Initialize the previous hidden state to zero
for every time-step $t$ do
    $a^t = f(\sum_{i=1}^{n} W_{.i} x_i^t + \sum_{i=1}^{h} H_{.i} a_i^{t-1})$        ▷ $a_t$ is the new state
    $o^t = g(\sum_{j=1}^{h} U_{.j} a_j^t)$
```

- $a_1, \ldots, a_T$ is the sequence of hidden states produced

Recurrent Neural Network: Forward Propagation

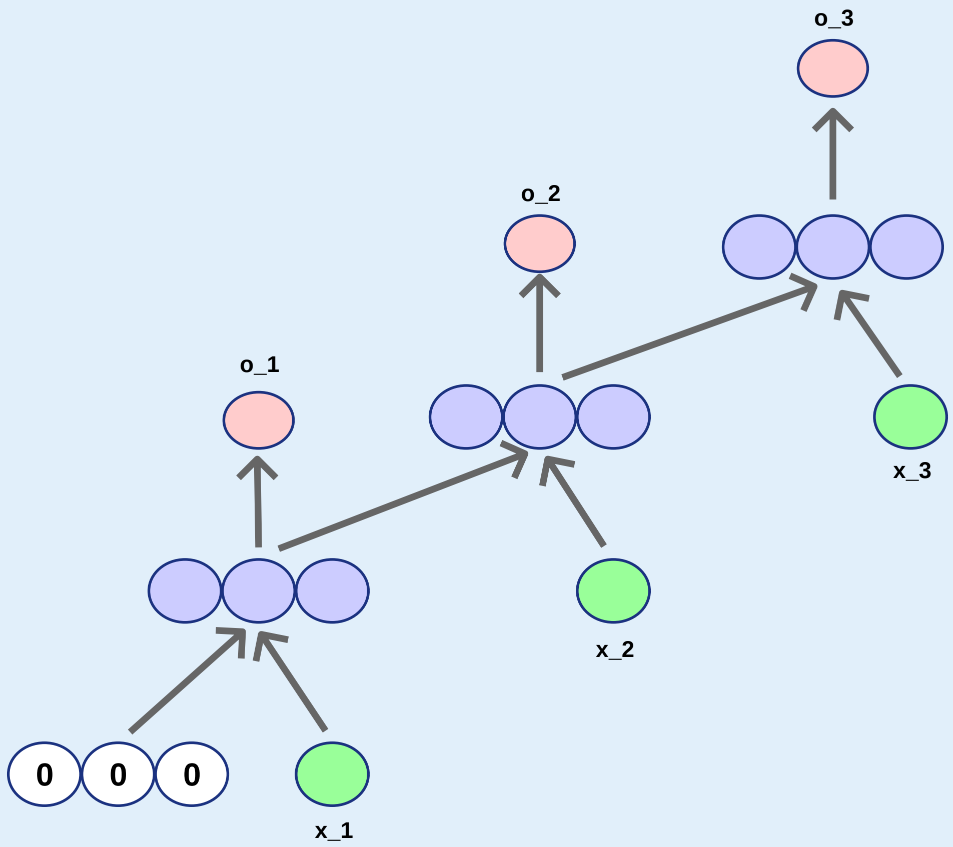


Figure: Forward Propagation RNN

Recurrent Neural Network: Forward Propagation

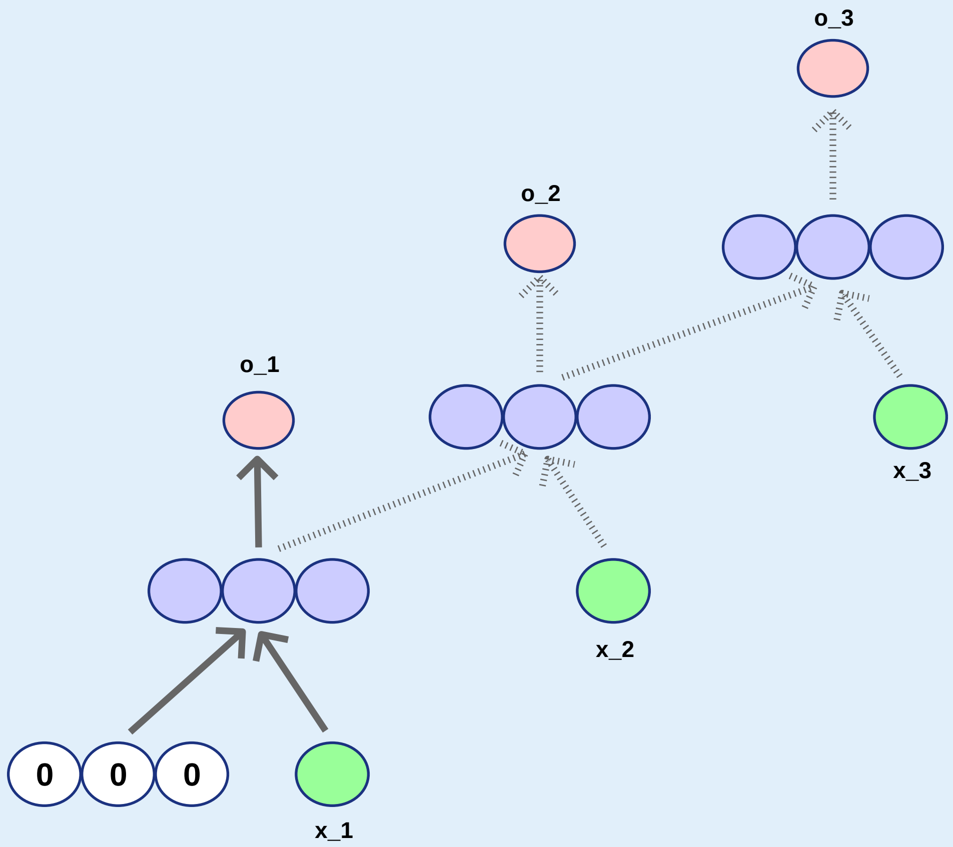


Figure: Forward Propagation RNN: Time-Step 1

Recurrent Neural Network: Forward Propagation

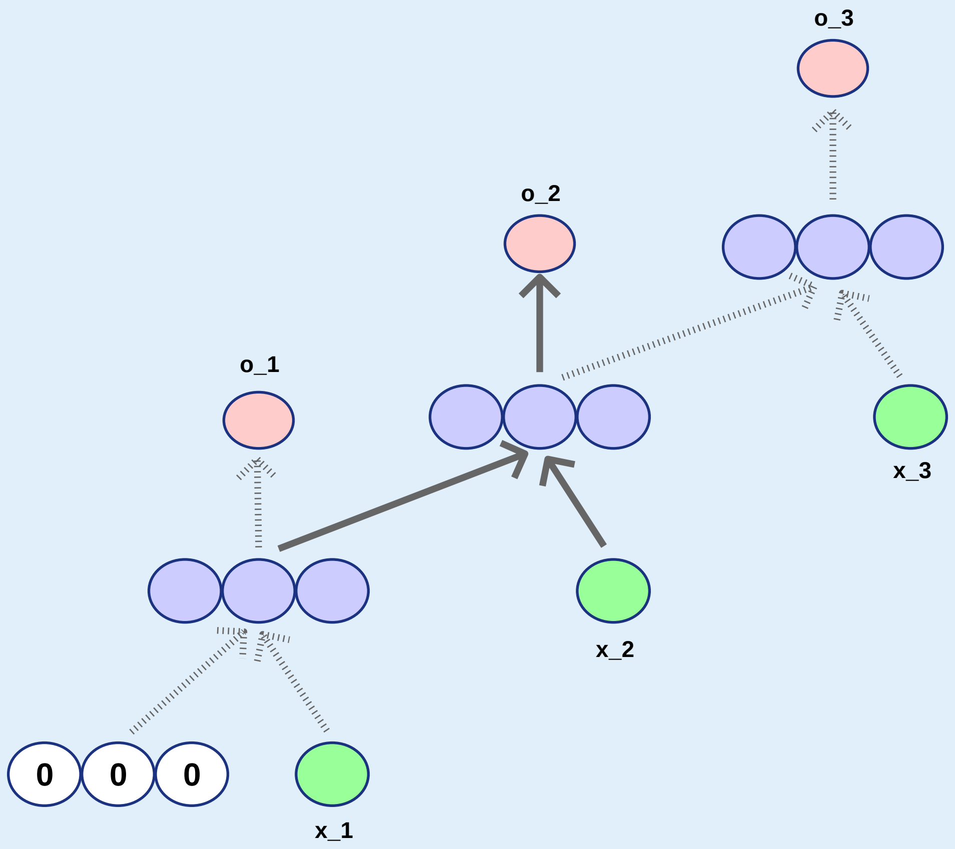


Figure: Forward Propagation RNN: Time-Step 2

Recurrent Neural Network: Forward Propagation

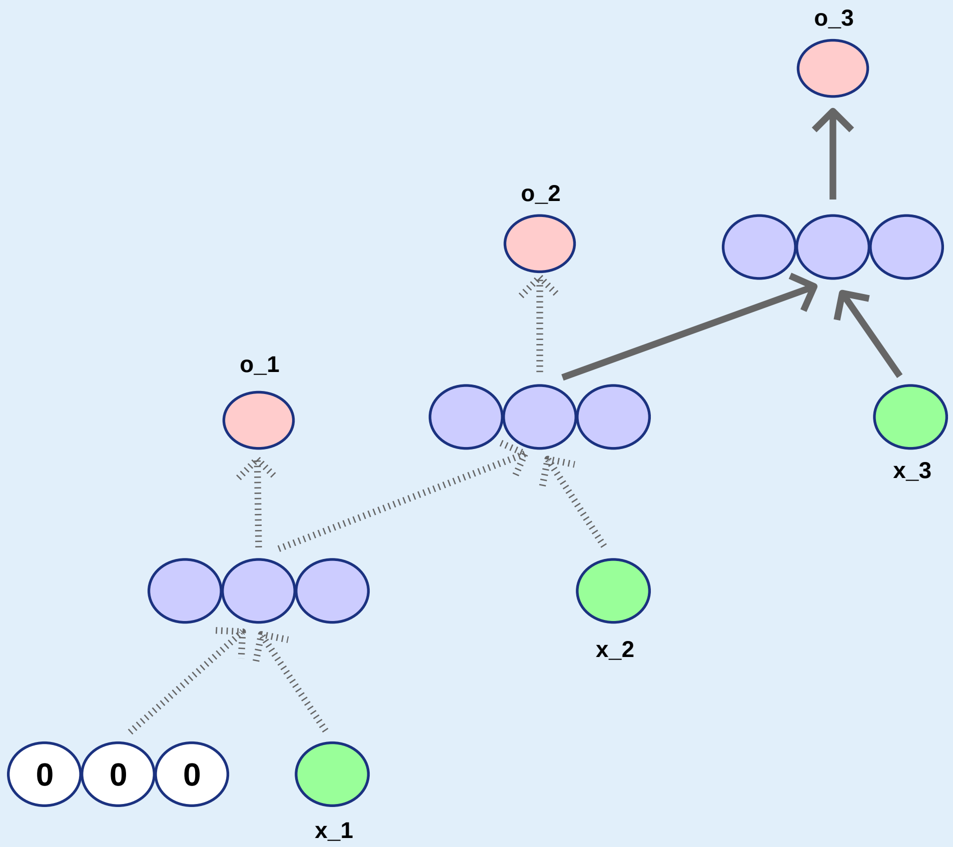


Figure: Forward Propagation RNN: Time-Step 3

Recurrent Neural Network: Backpropagation Through Time(BPTT) [Werbos 1990]

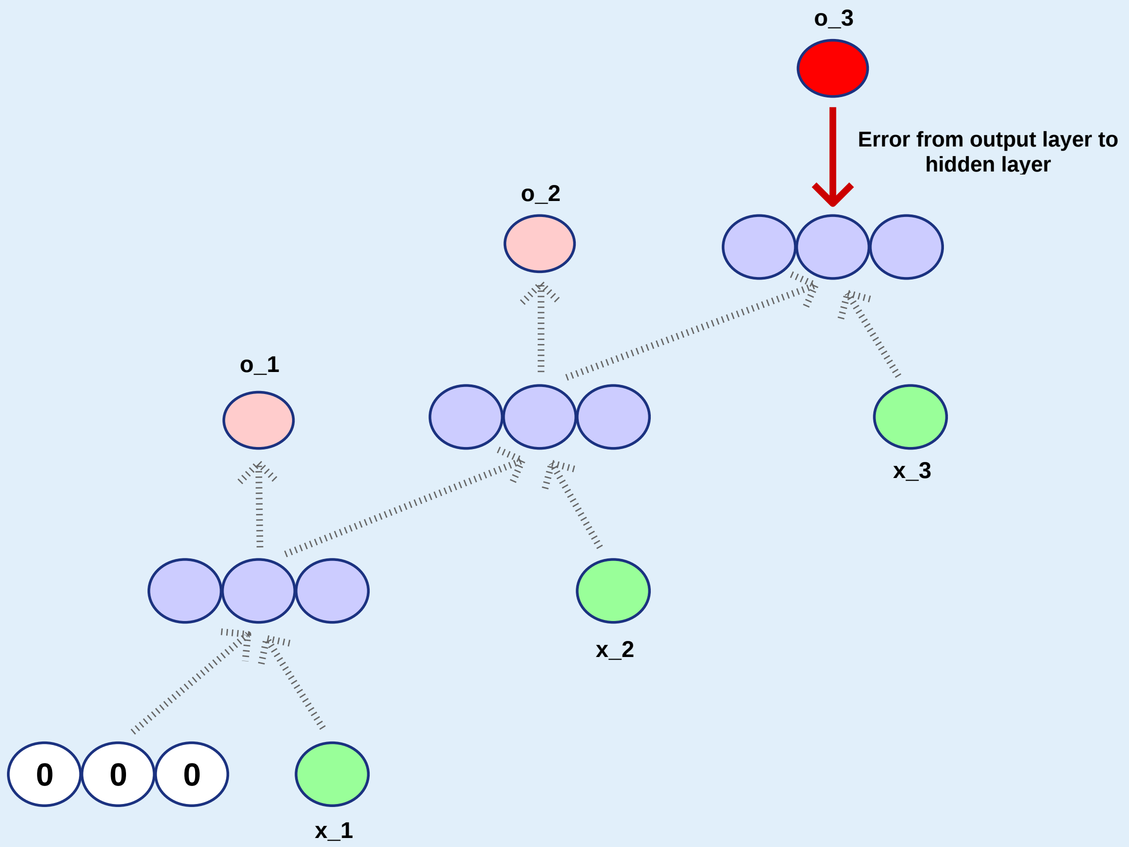


Figure: RNN Backpropagation Through Time: Time-Step 3

Recurrent Neural Network: Backpropagation Through Time(BPTT) [Werbos 1990]

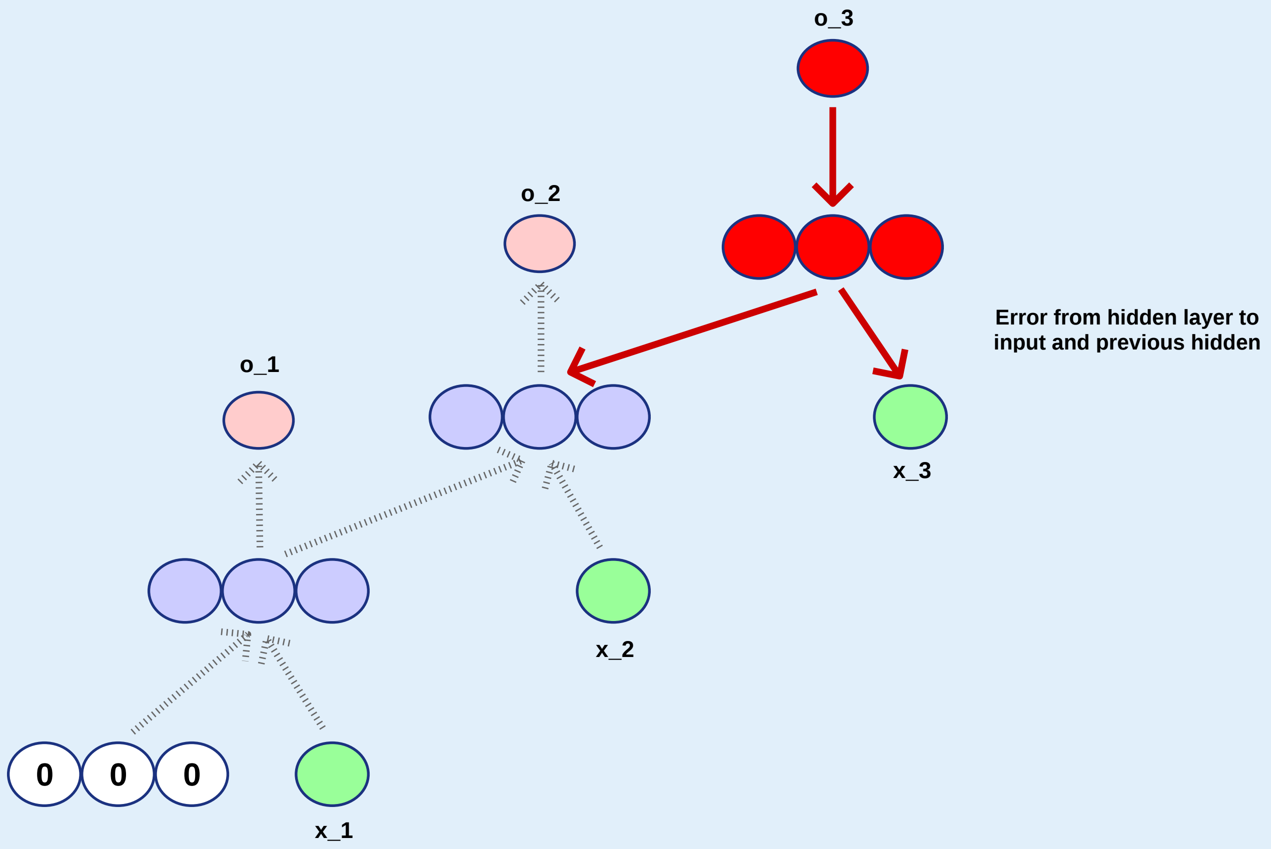


Figure: RNN Backpropagation Through Time: Time-Step 3

Recurrent Neural Network: Backpropagation Through Time(BPTT) [Werbos 1990]

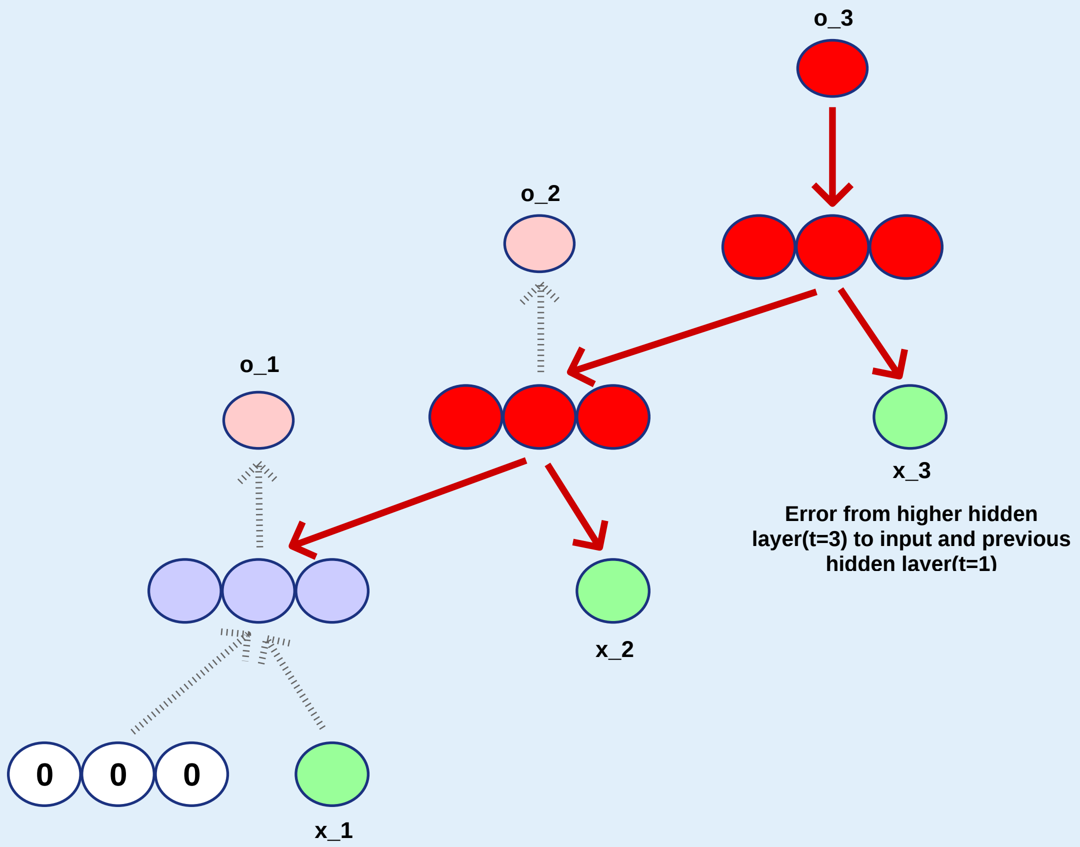

o_3
o_2
o_1
x_3
Error from higher hidden layer(t=3) to input and previous hidden layer(t=1)
x_2
0
0
0
x_1

Figure: RNN Backpropagation Through Time: Time-Step 3

Recurrent Neural Network: Backpropagation Through Time(BPTT) [Werbos 1990]

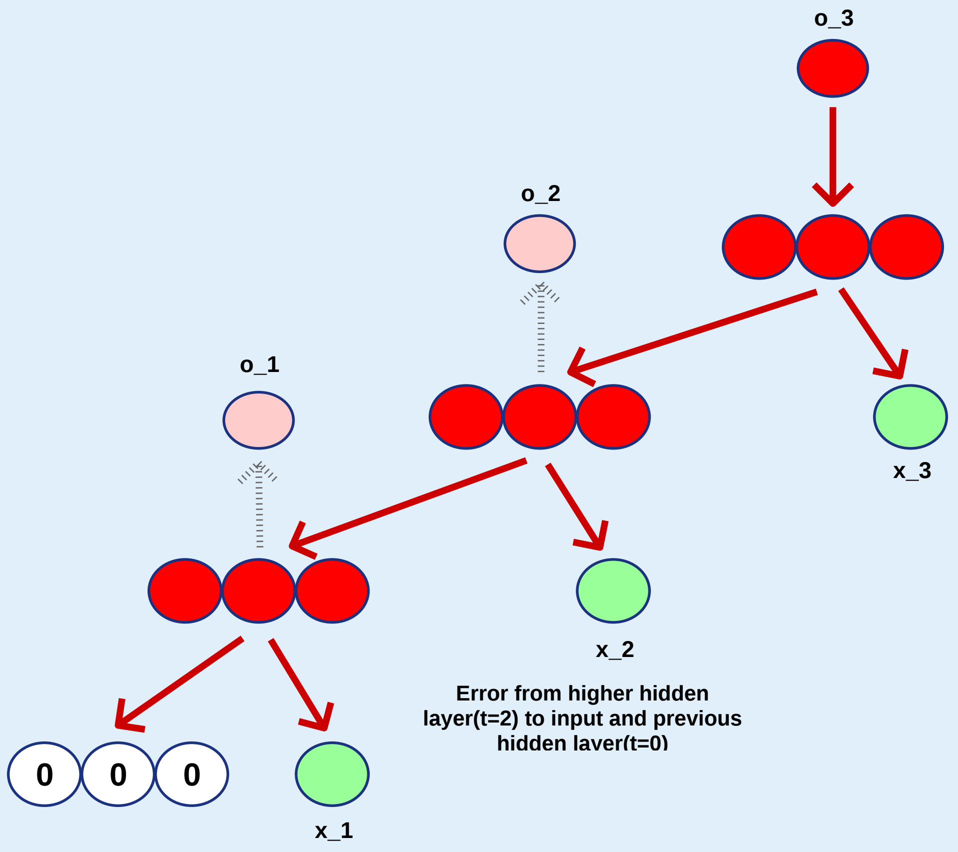

o_3
o_2
o_1
x_3
x_2
x_1
0
0
0
Error from higher hidden layer(t=2) to input and previous hidden layer(t=0)

Figure: RNN Backpropagation Through Time: Time-Step 3

Recurrent Neural Network: Backpropagation Through Time(BPTT) [Werbos 1990]

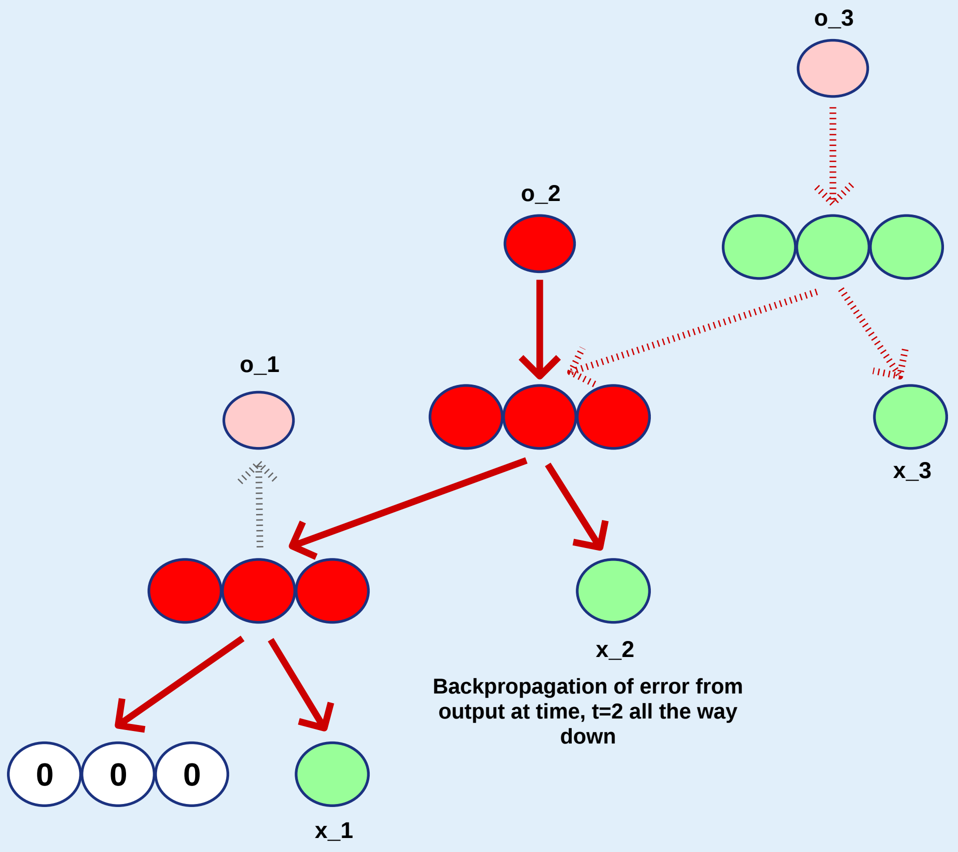

o_3
o_2
o_1
x_3
x_2
x_1
0
0
0
Backpropagation of error from output at time, t=2 all the way down

Figure: RNN Backpropagation Through Time: Time-Step 2

Deep Neural Network

Deep v/s Shallow Architectures

- Deep networks can learn complex function with relatively less number of parameters compared to shallow ones [Bengio 2009]
- Many problems tend to be solved in a hierarchical way (*Example: Image Classification*)

Plan

Challenges in training Neural Networks

- Non-convex Optimization (non-linear activations)
- Vanishing Gradient Problem
- Exploding Gradient Problem

Vanishing Gradient Problem [**Bengio:1994**]

- Presence of non-linear activations at every layer makes gradient vanish down the line
- Because of diluted gradient, the parameters will be close to their random initialization values at the lower layers

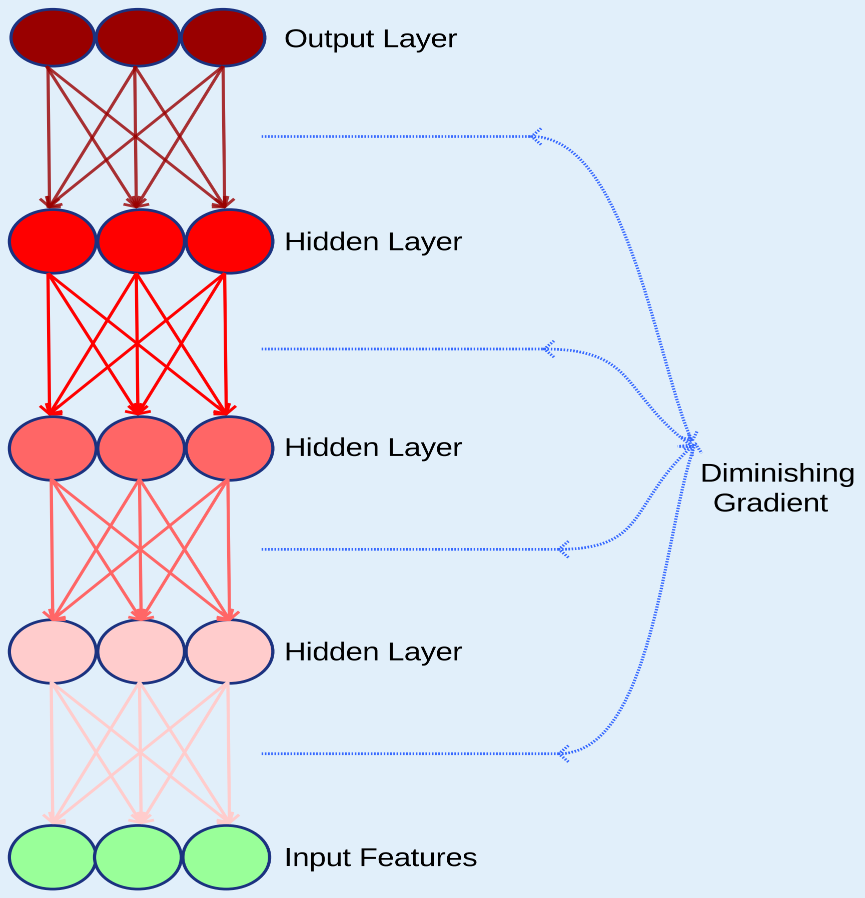
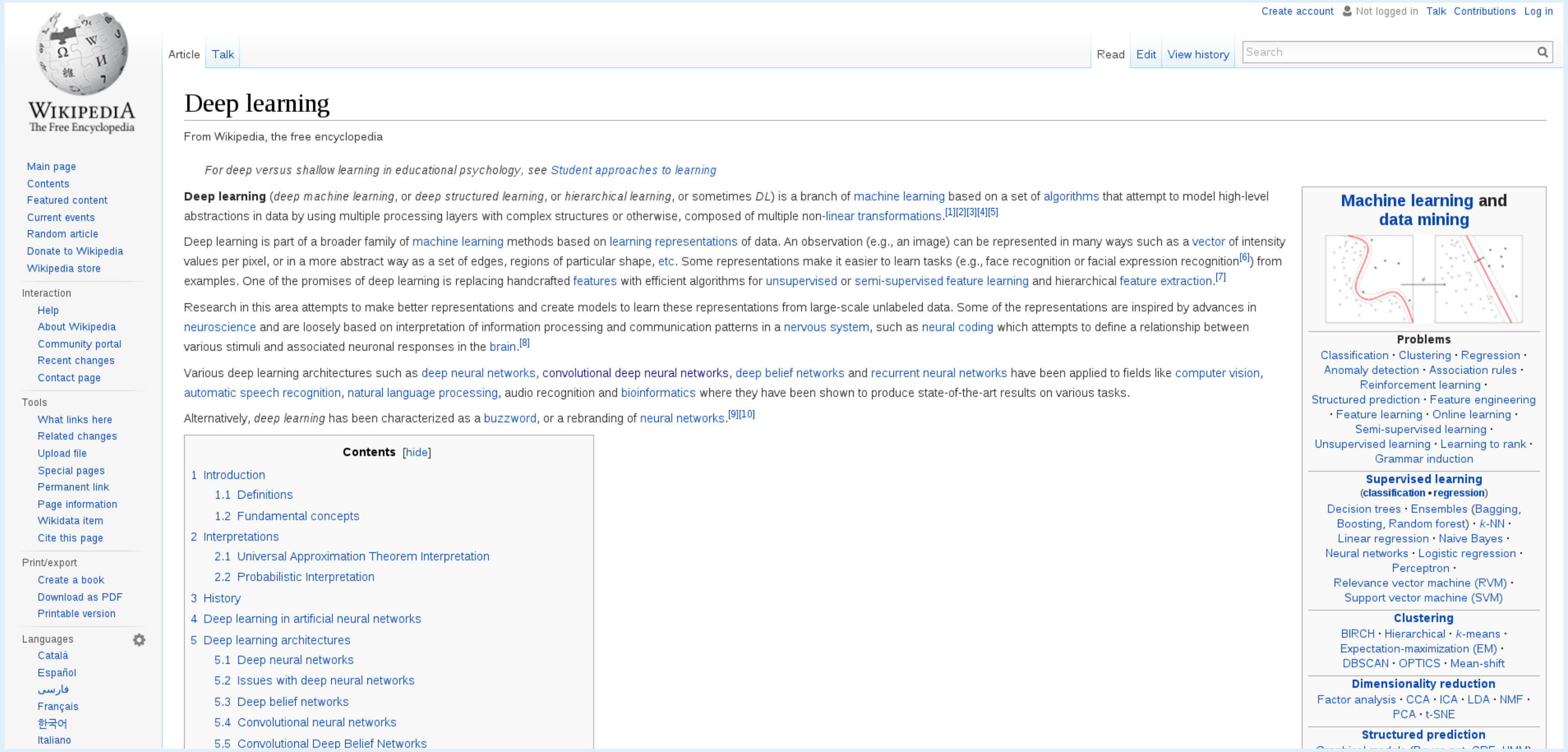


Figure: Backpropagation: Vanishing Gradient Problem

Unsupervised Feature Learning

Can we learn features from Unlabeled Data?



Create account · Not logged in · Talk · Contributions · Log in

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Català
Español
فارسی
Français
한국어
Italiano

Article · Talk · Read · Edit · View history · Search

Deep learning

From Wikipedia, the free encyclopedia

For deep versus shallow learning in educational psychology, see Student approaches to learning

Deep learning (*deep machine learning*, or *deep structured learning*, or *hierarchical learning*, or sometimes *DL*) is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers with complex structures or otherwise, composed of multiple non-linear transformations.[1][2][3][4][5]

Deep learning is part of a broader family of machine learning methods based on learning representations of data. An observation (e.g., an image) can be represented in many ways such as a vector of intensity values per pixel, or in a more abstract way as a set of edges, regions of particular shape, etc. Some representations make it easier to learn tasks (e.g., face recognition or facial expression recognition[6]) from examples. One of the promises of deep learning is replacing handcrafted features with efficient algorithms for unsupervised or semi-supervised feature learning and hierarchical feature extraction.[7]

Research in this area attempts to make better representations and create models to learn these representations from large-scale unlabeled data. Some of the representations are inspired by advances in neuroscience and are loosely based on interpretation of information processing and communication patterns in a nervous system, such as neural coding which attempts to define a relationship between various stimuli and associated neuronal responses in the brain.[8]

Various deep learning architectures such as deep neural networks, convolutional deep neural networks, deep belief networks and recurrent neural networks have been applied to fields like computer vision, automatic speech recognition, natural language processing, audio recognition and bioinformatics where they have been shown to produce state-of-the-art results on various tasks.

Alternatively, *deep learning* has been characterized as a buzzword, or a rebranding of neural networks.[9][10]

Contents [hide]

Machine learning and data mining

Problems
Classification · Clustering · Regression · Anomaly detection · Association rules · Reinforcement learning · Structured prediction · Feature engineering · Feature learning · Online learning · Semi-supervised learning · Unsupervised learning · Learning to rank · Grammar induction

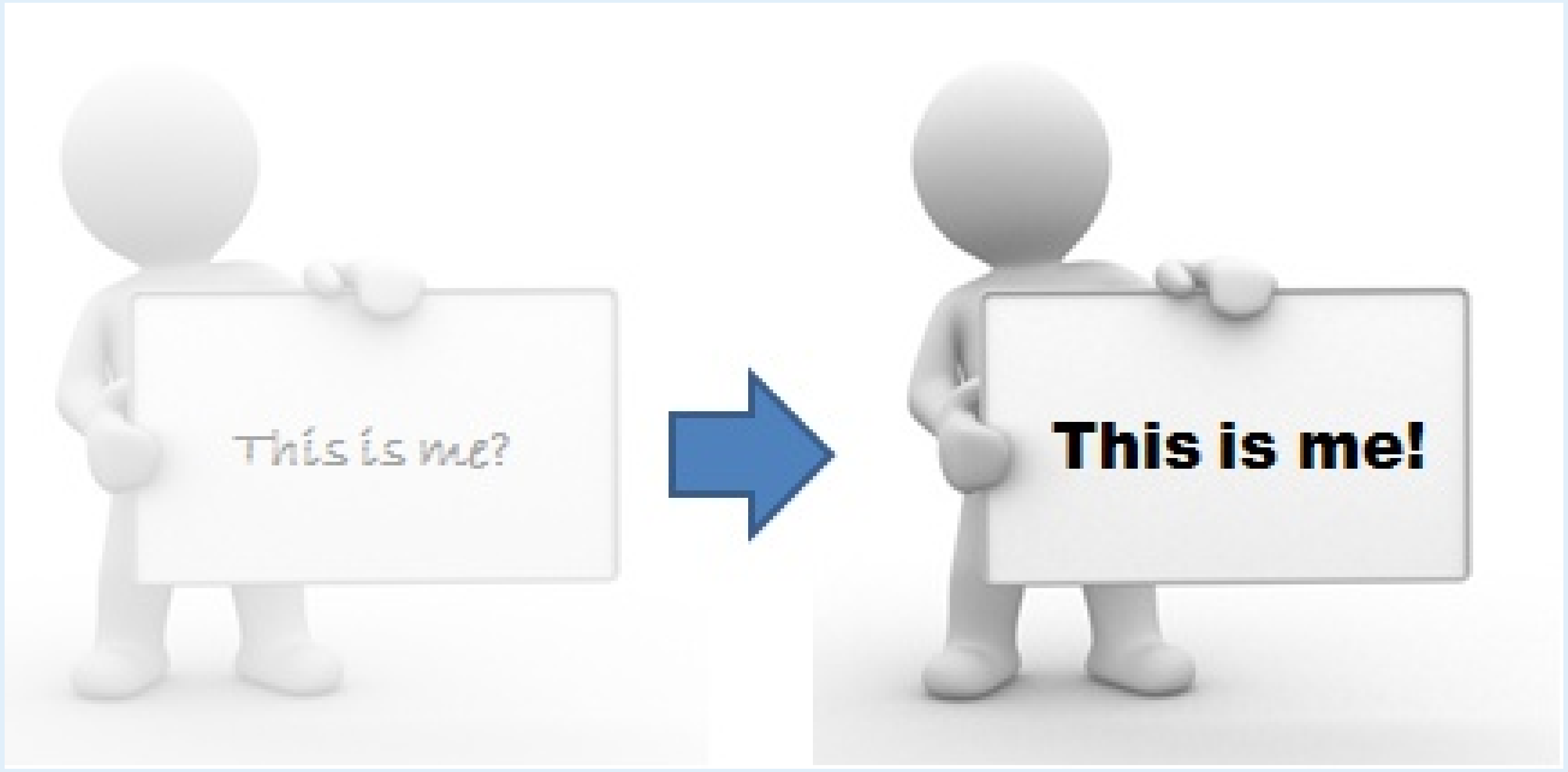
Supervised learning
(**classification** • **regression**)
Decision trees · Ensembles (Bagging, Boosting, Random forest) · *k*-NN · Linear regression · Naive Bayes · Neural networks · Logistic regression · Perceptron · Relevance vector machine (RVM) · Support vector machine (SVM)

Clustering
BIRCH · Hierarchical · *k*-means · Expectation-maximization (EM) · DBSCAN · OPTICS · Mean-shift

Dimensionality reduction
Factor analysis · CCA · ICA · LDA · NMF · PCA · t-SNE

Structured prediction

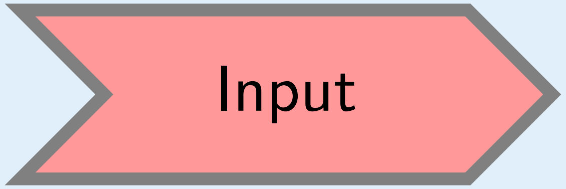
Understand Yourself [2]



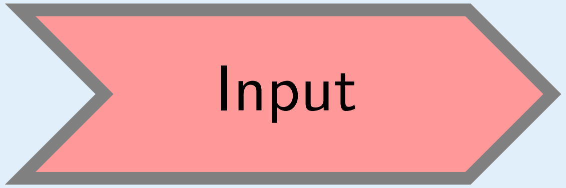
[2]http://www.humandesign.ca/

Unsupervised Feature Learning

- Supervised Learning

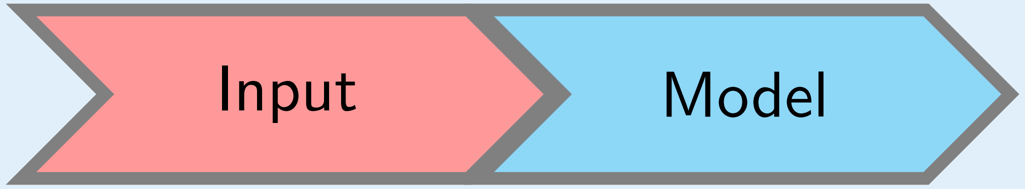


- Unsupervised Feature Learning

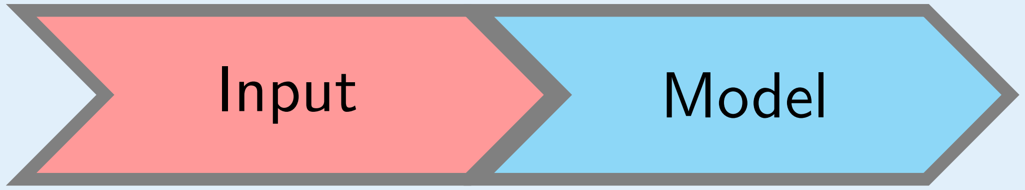


Unsupervised Feature Learning

- Supervised Learning

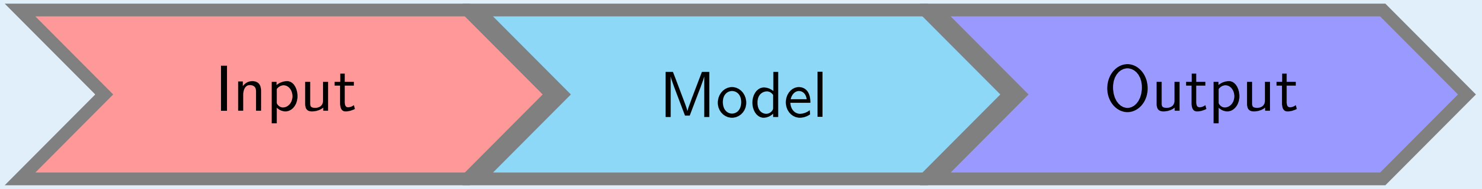


- Unsupervised Feature Learning

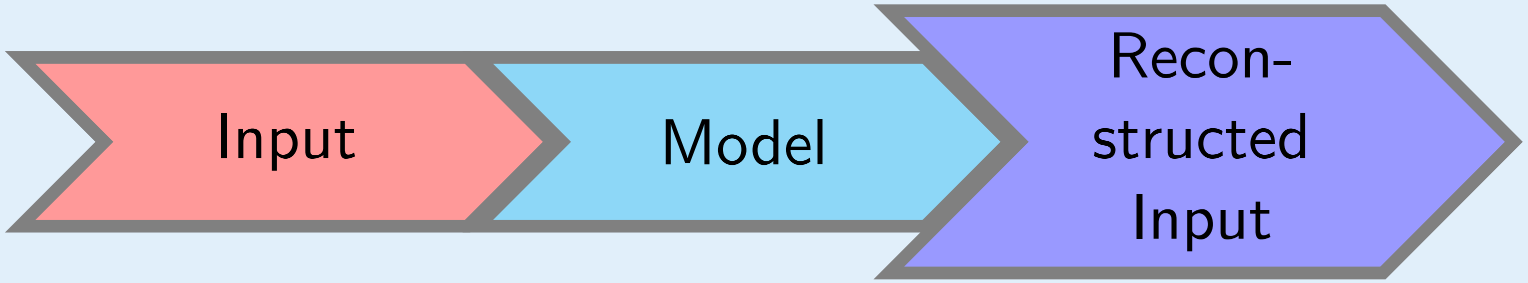


Unsupervised Feature Learning

- Supervised Learning



- Unsupervised Feature Learning



Unsupervised Feature Learning: AutoEncoder [G E Hinton and Salakhutdinov 2006]

- Train the network to reconstruct the input
- Encoder tries to extract relevant information from the data
- Decoder tries to generate the data using extracted features
- Care to be taken so not to let network learn an Identity function

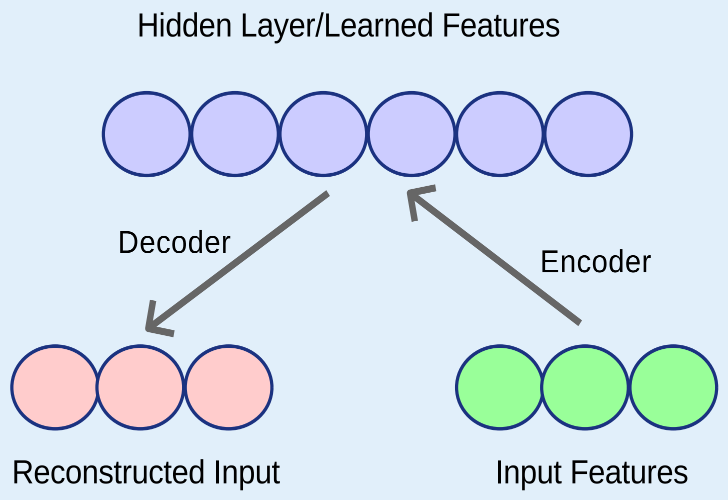


Figure: Auto-Encoder Architecture

AutoEncoders [Bengio et al. 2007]

- Given input x, $z = \sigma(Wx)$ be d-dimensional learned features
- Now reconstructed input from decoder, $y = f(Dz)$
- Train the network by minimizing some error
- For Mean Squared Error, $E = \frac{1}{2}\sum_{x \in D} \|y - x\|^2$

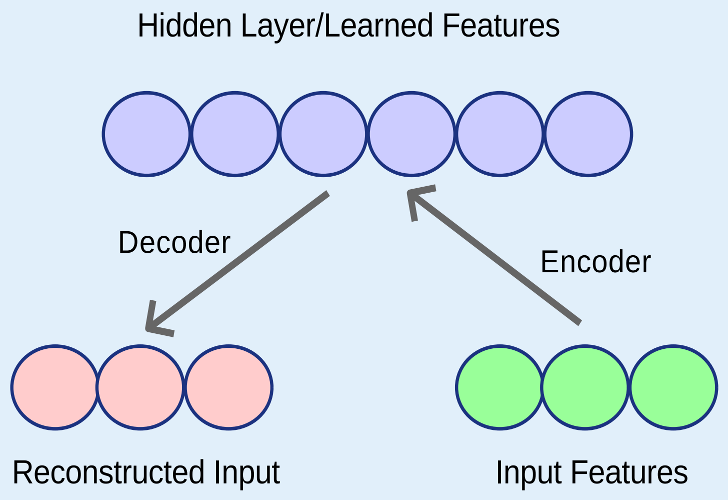


Figure: Auto-Encoder Architecture

Denoised AutoEncoders [Vincent et al. 2008]

- Given input x, add noise to the data to get $\hat{x}$
- Let $z = \sigma(W\hat{x})$ be d-dimensional learned features
- Now reconstructed input from decoder, $y = f(Dz)$
- Train the network by minimizing error
- For Mean Squared Error, $E = \frac{1}{2}\sum_{x \in D} \|y - x\|^2$

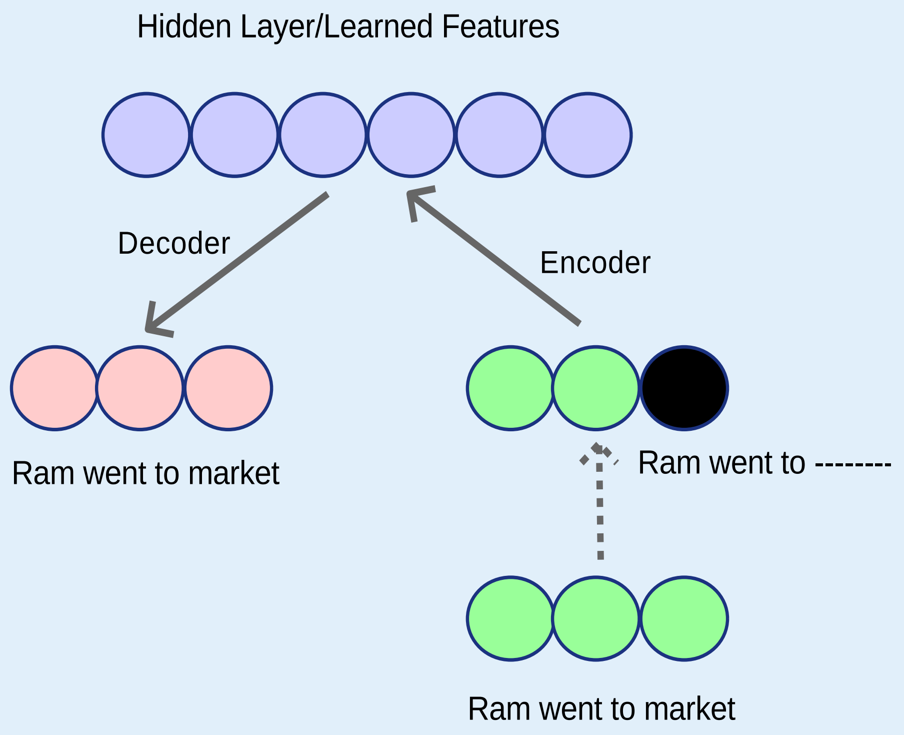


Figure: Denoised Auto-Encoder Architecture

Stacked AutoEncoders [Bengio et al. 2007]

- Train an Autoencoder to reconstruct input with one hidden layer

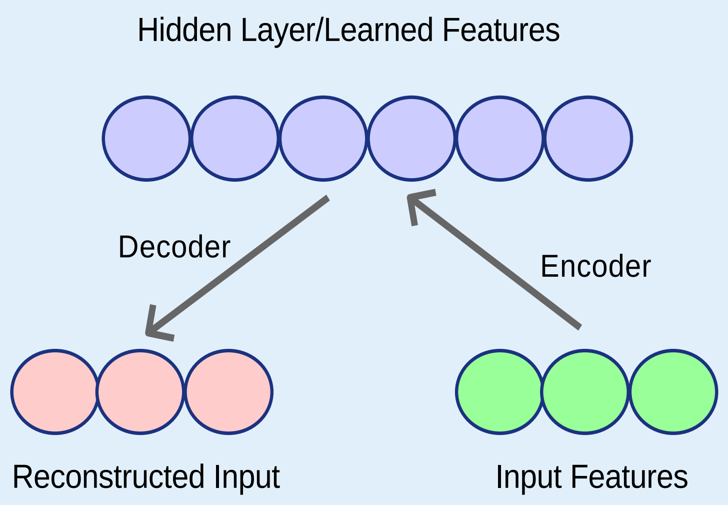


Figure: Unsupervised Pre-training: Stacked AutoEncoder

Stacked AutoEncoders [Bengio et al. 2007]

- Train an Autoencoder to reconstruct input with one hidden layer
- Use first hidden layer features as input and train an autoencoder on top of it

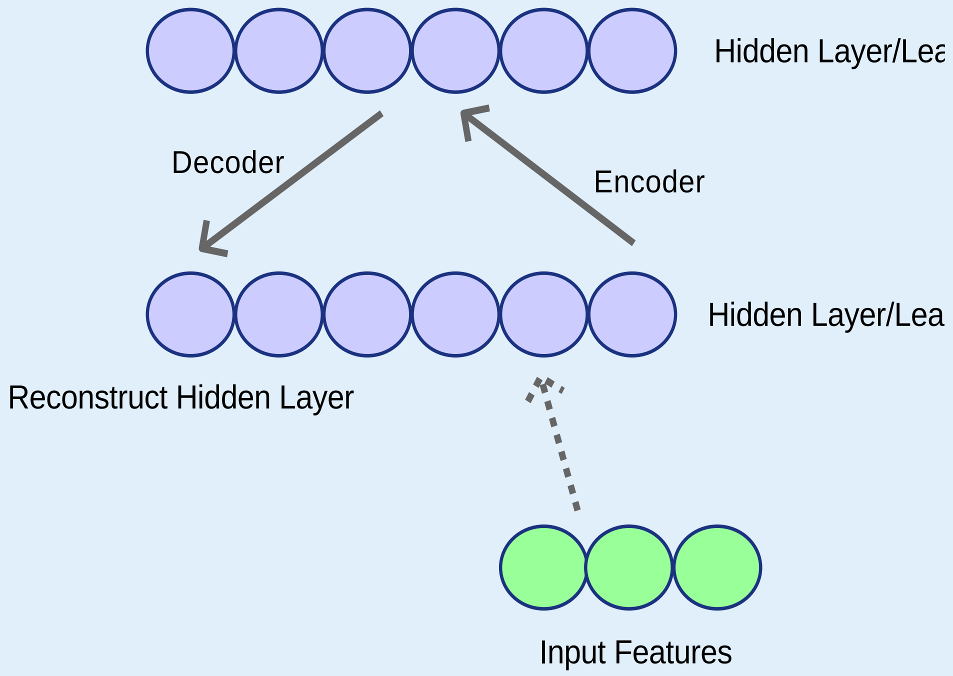


Figure: Unsupervised Pre-training: Stacked AutoEncoder

Stacked AutoEncoders [Bengio et al. 2007]

- Train an Autoencoder to reconstruct input with one hidden layer
- Use first hidden layer features as input and train an autoencoder on top of it
- Repeat this for many layers

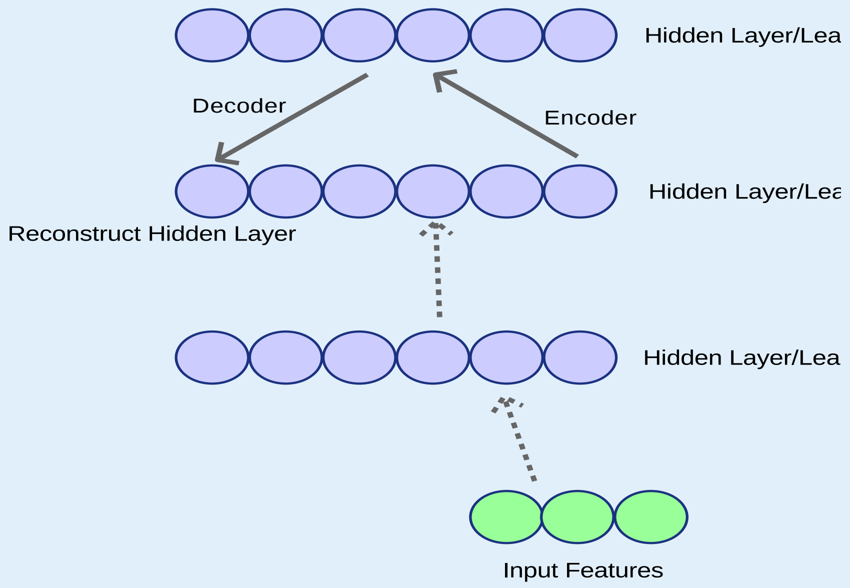


Figure: Unsupervised Pre-training: Stacked AutoEncoder

Stacked AutoEncoders [Bengio et al. 2007]

- Train an Autoencoder to reconstruct input with one hidden layer
- Use first hidden layer features as input and train an autoencoder on top of it
- Repeat this for many layers
- Discard all the decoder parameters and do supervised training

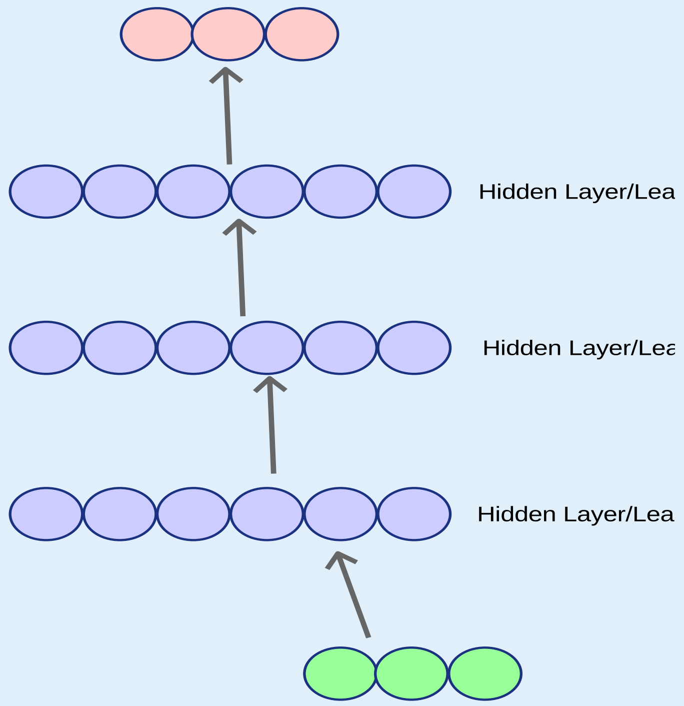


Figure: Fine Tuning Phase

Restricted Boltzmann Machines (RBMs) [Smolensky 1986]

- RBMs are Bipartite Undirected Graphical Models
- Two partitioning of the graph: visible nodes and hidden nodes
- Connection from input to hidden nodes are undirected
- No connection between visible nodes nor hidden nodes
- The visible units are binary units (can be extended to real or categorical values)

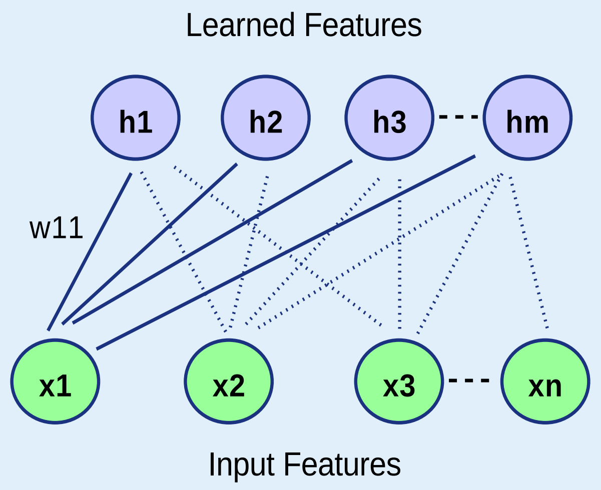


Figure: Restricted Boltzmann Machines (RBM)

Restricted Boltzmann Machines (RBMs) [Smolensky 1986]

- Let $X = (x_1, \ldots, x_n)$ be visible nodes
- Let $H = (h_1, \ldots, h_m)$ be hidden nodes
- Energy function for the joint assignment is given by

$$E(x, h) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} x_i h_j + \sum_{i=1}^{n} b_i x_i + \sum_{j=1}^{m} c_j h_j \tag{8}$$

- The probability of the joint assignment is given by,

$$p(x, h) = \frac{\exp^{-E(x,h)}}{Z} \tag{9}$$

- Here, Z is the partition function

Restricted Boltzmann Machines (RBMs) [Smolensky 1986]

- The objective is to maximize the likelihood of the data
- Only the visible units values are known

$$L(D) = \frac{1}{N} \sum_{x \in D} \log P(x) \tag{10}$$

$$L(D) = \frac{1}{N} \sum_{x \in D} \log \sum_{h \in H} P(x, h) \tag{11}$$

- The hidden node learns relevant features from the data
- The visible to hidden node connections be used to initialize a Feedforward Neural Network

Summary

- We began with a simple introduction to Perceptron Algorithm
- Failures of Perceptrons led to Multilayer Perceptron (Feed Forward Neural Network)
- Feed Forward Neural Network and Recurrent Neural Networks were introduced
- We looked at the challenges in training these networks
- Unsupervised Representation Learning algorithms which led to the success of Deep Learning were introduced
- We will now look at one architecture of Neural Network which was successfully applied on various tasks

References I

- Yoshua Bengio. “Learning Deep Architectures for AI”. In: *Found. Trends Mach. Learn.* 2.1 (Jan. 2009), pp. 1–127. ISSN: 1935-8237. DOI: 10.1561/2200000006. URL: http://dx.doi.org/10.1561/2200000006.
- Yoshua Bengio et al. “Greedy layer-wise training of deep networks”. In: *In NIPS*. MIT Press, 2007.
- Jeffrey L. Elman. “Finding structure in time”. In: *COGNITIVE SCIENCE* 14.2 (1990), pp. 179–211.

References II

- G E Hinton and R R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786 (July 2006), pp. 504–507. DOI: 10.1126/science.1127647. URL: http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google.
- Honglak Lee et al. *Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations*.
- F. Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory). Spartan Books, 1962. URL: https://books.google.co.in/books?id=7FhRAAAAMAAJ.

References III

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. Ed. by David E. Rumelhart and James L. Mcclelland. Cambridge, MA: MIT Press, 1986, pp. 318–362.

References IV

P. Smolensky. “Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1”. In: ed. by David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group. Cambridge, MA, USA: MIT Press, 1986. Chap. Information Processing in Dynamical Systems: Foundations of Harmony Theory, pp. 194–281. ISBN: 0-262-68053-X. URL: http://dl.acm.org/citation.cfm?id=104279.104290.

Pascal Vincent et al. *Extracting and Composing Robust Features with Denoising Autoencoders*. 2008.

P.J. Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78.10 (Oct. 1990), pp. 1550–1560. ISSN: 0018-9219. DOI: 10.1109/5.58337.

Long Short-Term Memory Models

Prerana Singhal

IIT Bombay

LSTM :: Overview

- A **modified version of Recurrent Neural Network** :: more complex
- Deals with the limitations of RNN ::
  - — Vanishing Gradient (over the time steps)
  - — Exploding Gradient (over the time steps)
- This model is an attempt to allow the unit activations to retain important information over a much longer period of time than the traditional RNN.
- **An LSTM network is well-suited to learn from experience to classify, process and predict time series when there are very long time lags of unknown size between important events.** (wikipedia)

LSTM :: Long Term ? and Short Term ?

Long Short Term Memory Model :: Information is stored in two distinct ways

- The **activations** of the units are a function of the recent history of the model, and so form a **short-term memory**.

  — *Much like knowing when you are hungry each time of the day*

- The **weights** too form a memory, called a **long-term memory**, as they are modified based on experience, but the timescale of the weight change is much slower than that of the activations.

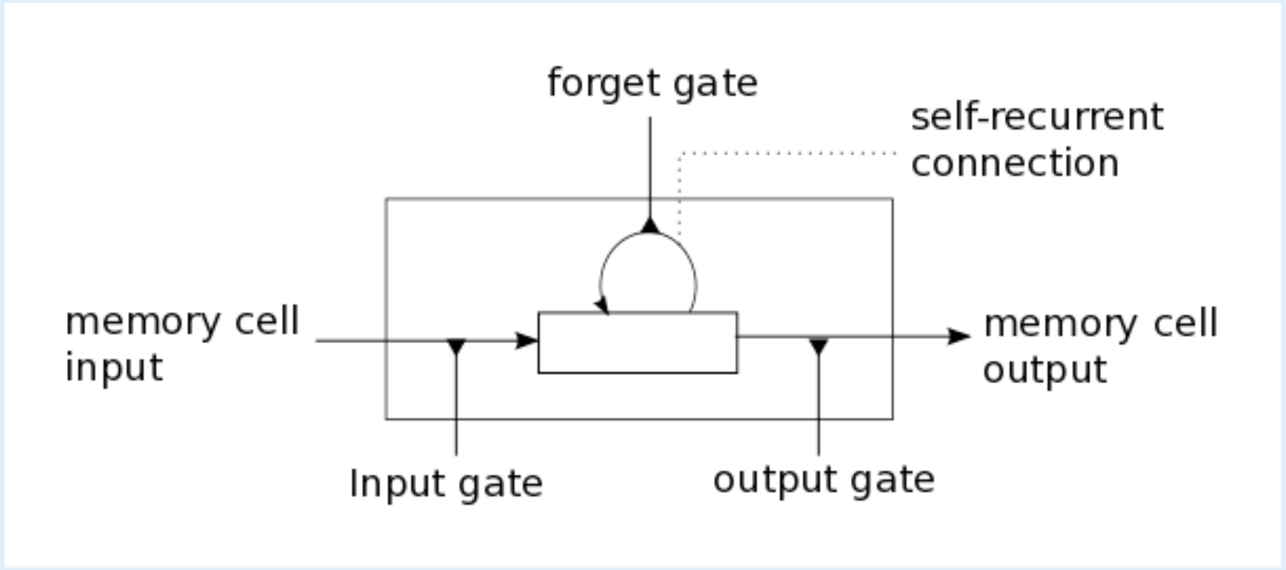
  — *Much like knowing when you know you are getting too fat or too thin*

LSTM :: Main Motivation

Memory Cell :: A new structure introduced composed of four main elements:

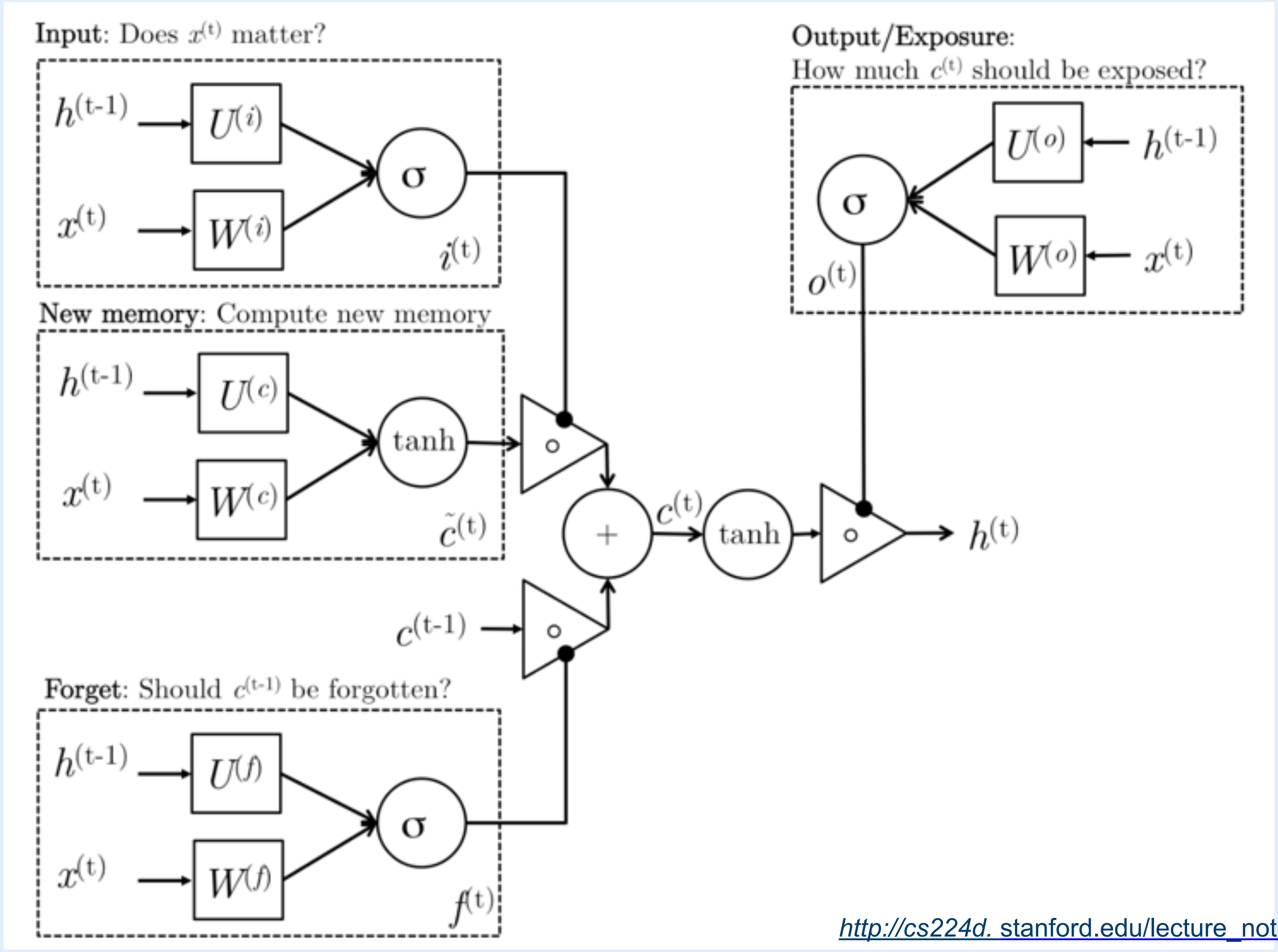
- an input gate,
- a neuron with a self-recurrent connection (a connection to itself),
- a forget gate,
- an output gate.

The gates serve to modulate the interactions between the memory cell itself and its environment.



http://deeplearning.net/tutorial/lstm.html

LSTM :: Architecture



http://cs224d. stanford.edu/lecture_notes/LectureNotes4.pdf

LSTM :: Architecture

Mathematical Formulation of LSTM Units ::

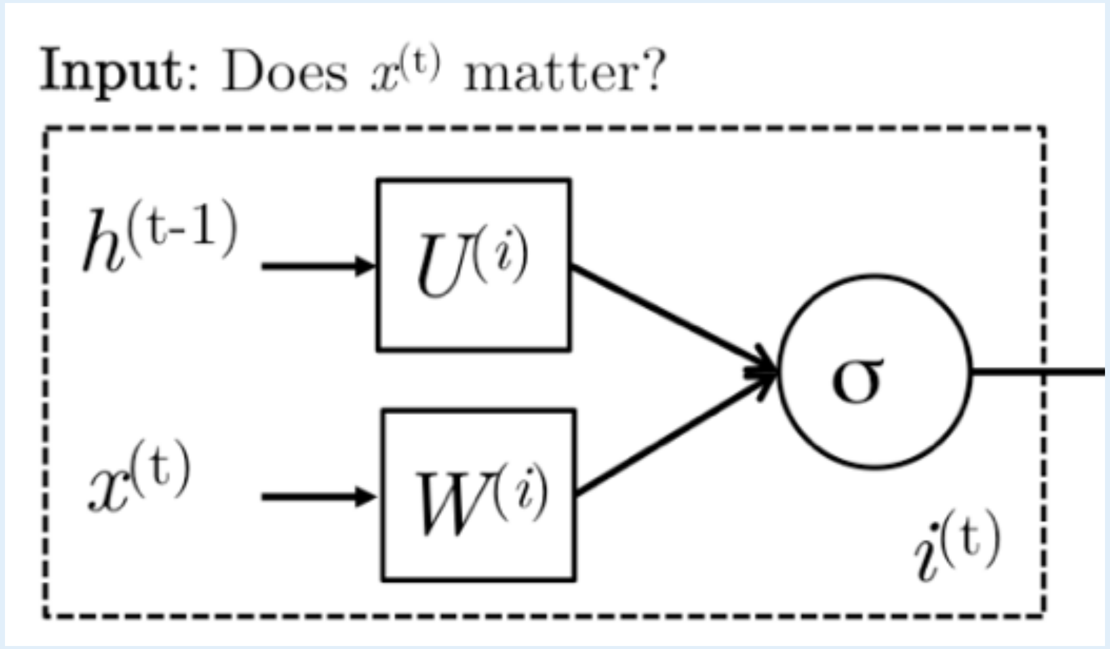
$$
\begin{aligned}
i^{(t)} &= \sigma(W^{(i)}x^{(t)} + U^{(i)}h^{(t-1)}) && \text{(Input gate)} \\
f^{(t)} &= \sigma(W^{(f)}x^{(t)} + U^{(f)}h^{(t-1)}) && \text{(Forget gate)} \\
o^{(t)} &= \sigma(W^{(o)}x^{(t)} + U^{(o)}h^{(t-1)}) && \text{(Output/Exposure gate)} \\
\tilde{c}^{(t)} &= \tanh(W^{(c)}x^{(t)} + U^{(c)}h^{(t-1)}) && \text{(New memory cell)} \\
c^{(t)} &= f^{(t)} \circ \tilde{c}^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{(Final memory cell)} \\
h^{(t)} &= o^{(t)} \circ \tanh(c^{(t)})
\end{aligned}
$$

LSTM :: Essential Components (1/5)

1. **Input Gate** :

   — Uses the input word and the past hidden state to determine whether or not the input is worth preserving

Much like deciding whether or not this sentence right here is worth remembering

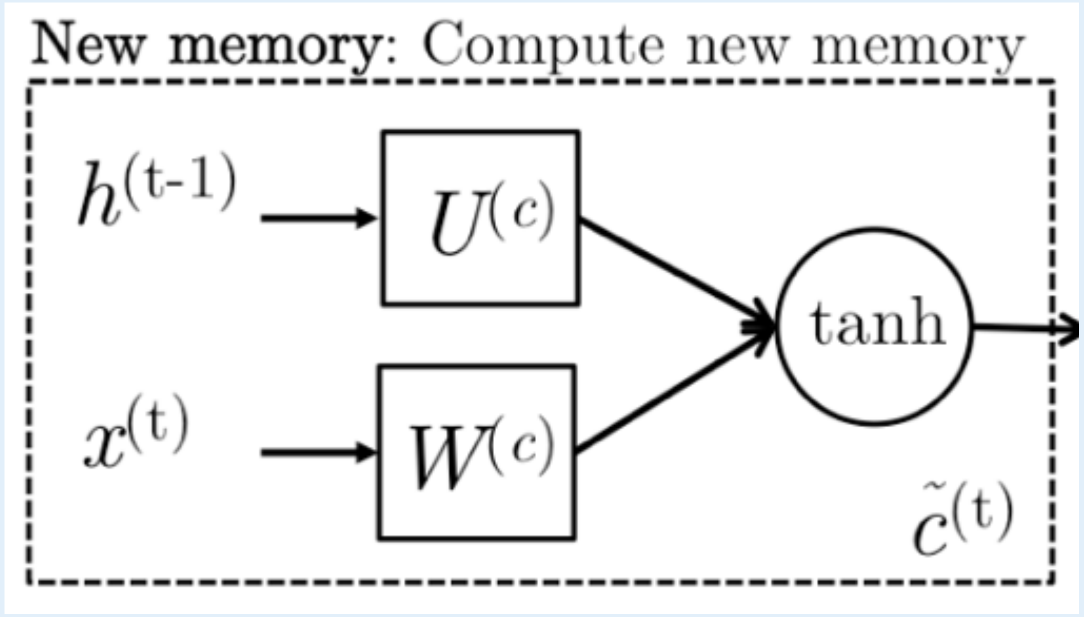


LSTM :: Essential Components (2/5)

2. **New Memory Generation** :

— Uses the input word and the past hidden state to generate a new memory which includes aspects of the new word

Much like understanding that this sentence is for fun based on this sentence and the one encountered in the previous slide

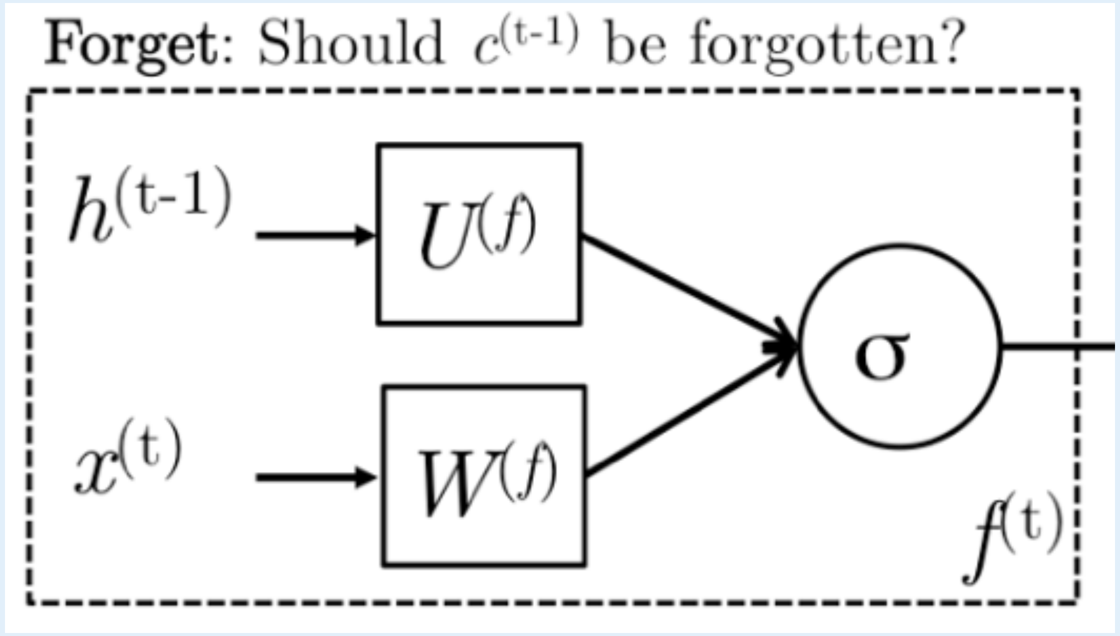


LSTM :: Essential Components (3/5)

3. **Forget Gate** :

— Uses the input word and the past hidden state to make an assessment on whether the past memory cell is useful for the computation of the current memory cell

Much like knowing that this sentence is not worth paying attention to based on just by reading this and hence italic sentences encountered in previous slides can be forgotten



LSTM :: Essential Components (4/5)

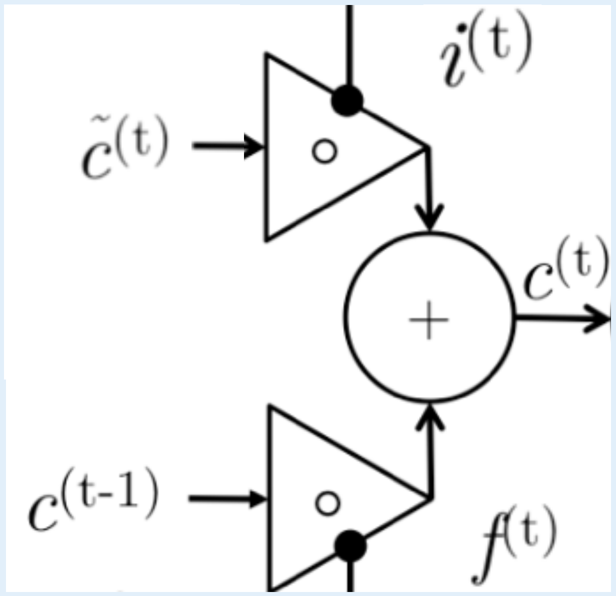
4. **Final Memory Generation** :

   — First takes the advice of the forget gate and accordingly forgets the past memory.

   — Then takes the advice of the input gate and accordingly gates the new memory.

   — Then sums these two results to produce the final memory.

   Much like deciding that it is OK to overlook these italicised sentences with no actual technical information (or is it?)

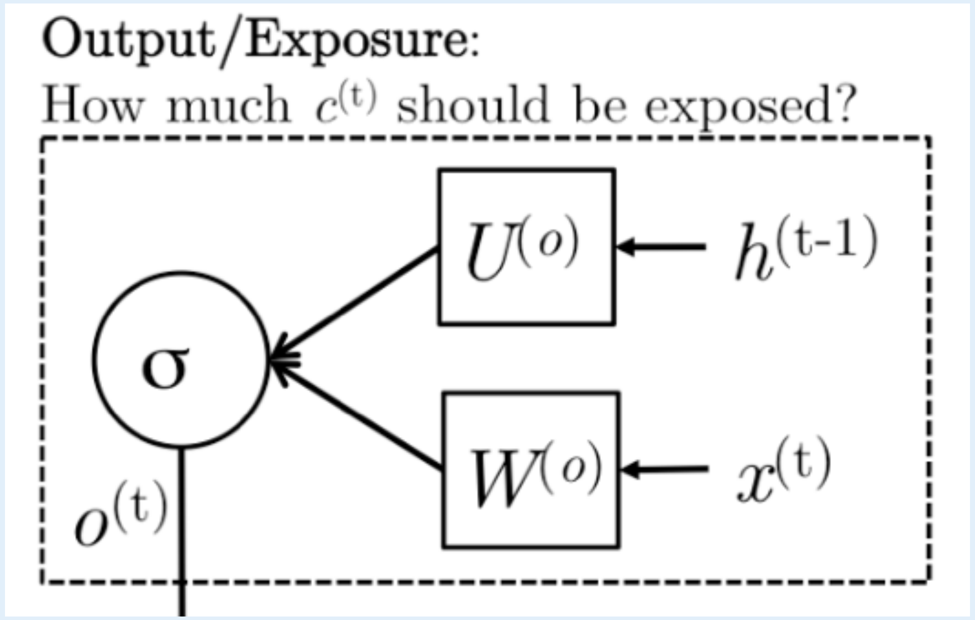


LSTM :: Essential Components (5/5)

5. **Output Gate** :

    — Makes the assessment regarding what parts of the memory needs to be exposed/present in the hidden state

Much like thinking right now whether to retell this sentence right here (or any other part of this presentation for that matter) when sharing this tutorial's take-away with others



LSTM :: Summary

- LSTM model looks very complicated, feels very advanced in architecture, and seems to be very effective to solve problems
- In fact, **an LSTM network is universal in the sense that given enough network units, it can compute anything a conventional computer can compute, provided it has the proper weight matrix, which may be viewed as its program.** (wikipedia)
- Demands :: enough training data, enough training time
- Promises :: Learn from experience over long time and try to give as accurate a prediction for the problem at hand as possible

LSTM :: References

- S. Hochreiter and J. Schmidhuber, "**Long short-term memory**," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- "**Lstm networks for sentiment analysis**." *http://deeplearning.net/tutorial/ lstm.html.*
- L. Deng and D. Yu, "**Deep learning: methods and applications,**" *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.
- Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "**A neural probabilistic language model**," *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.
- F.A.Gers, J.Schmidhuber, and F. Cummins,"**Learning to forget : Continual prediction with lstm**," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.
- J. R. Anderson, **Language, memory, and thought.** Psychology Press, 2013.
- R. M. Richard Socher, Milad Mohammadi, "*Deep learning for nlp*." http://cs224d. stanford.edu/lecture_notes/LectureNotes4.pdf
- A. Graves et al., **Supervised sequence labelling with recurrent neural networks**, vol. 385. Springer, 2012.
- I. Aleksander and H. Morton, **An introduction to neural computing**, vol. 3. Chapman & Hall London, 1990.

Convolutional Neural Networks

Prerana Singhal

IIT Bombay

CNN :: Overview

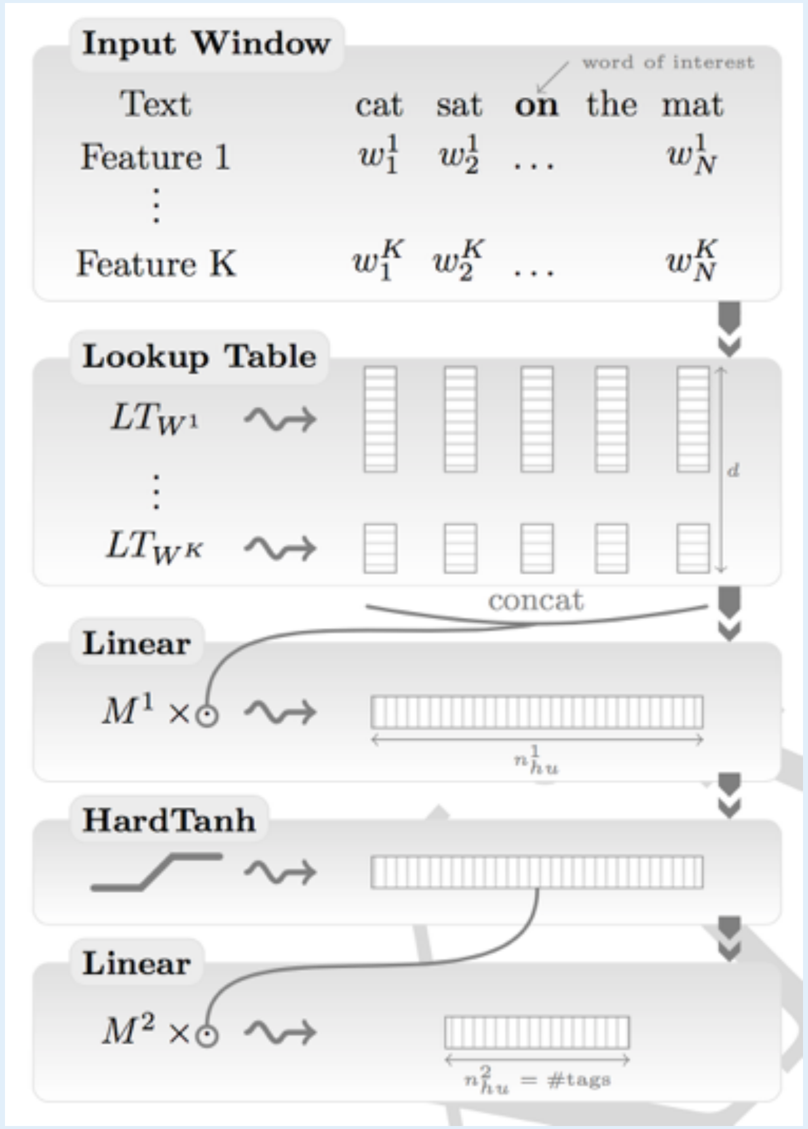
- **Convolutional neural network (CNN, or ConvNet)** is a type of feed-forward artificial neural network where the individual neurons are tiled in such a way that they respond to overlapping regions in the input field. (wikipedia)
- Convolutional Neural Networks are basically biologically-inspired variants of Multi-Layer Perceptrons
- Architecture is complex :: Not fully connected
- CNN has wide effective applications in Image Processing Industry
  - — Has been found useful in the field of Natural Language Processing too
- Convolutional Filters learn good representations automatically, without needing to represent the whole vocabulary.

CNN :: Motivation

- The words are transformed into feature vectors which are basically **word embeddings** learnt by training the neural network.

- ***Window-approach network*** is not desirable because many a times,
  - — classification w.r.t. to a particular word depends on some far-away word in the sentence not falling inside the window boundaries.

- Hence, **sentence network approach** is preferred for various NLP tasks to consider the whole sentence for producing any output
  - — makes use of **Convolution Neural Network**



CNN :: Motivation

Convolution Neural Networks are based on the following principles ::

- Local Receptive Fields (small windows of texts which are overlapping)
- Shared Weights (over windows)
- Pooling (or down-sampling), mainly max pooling

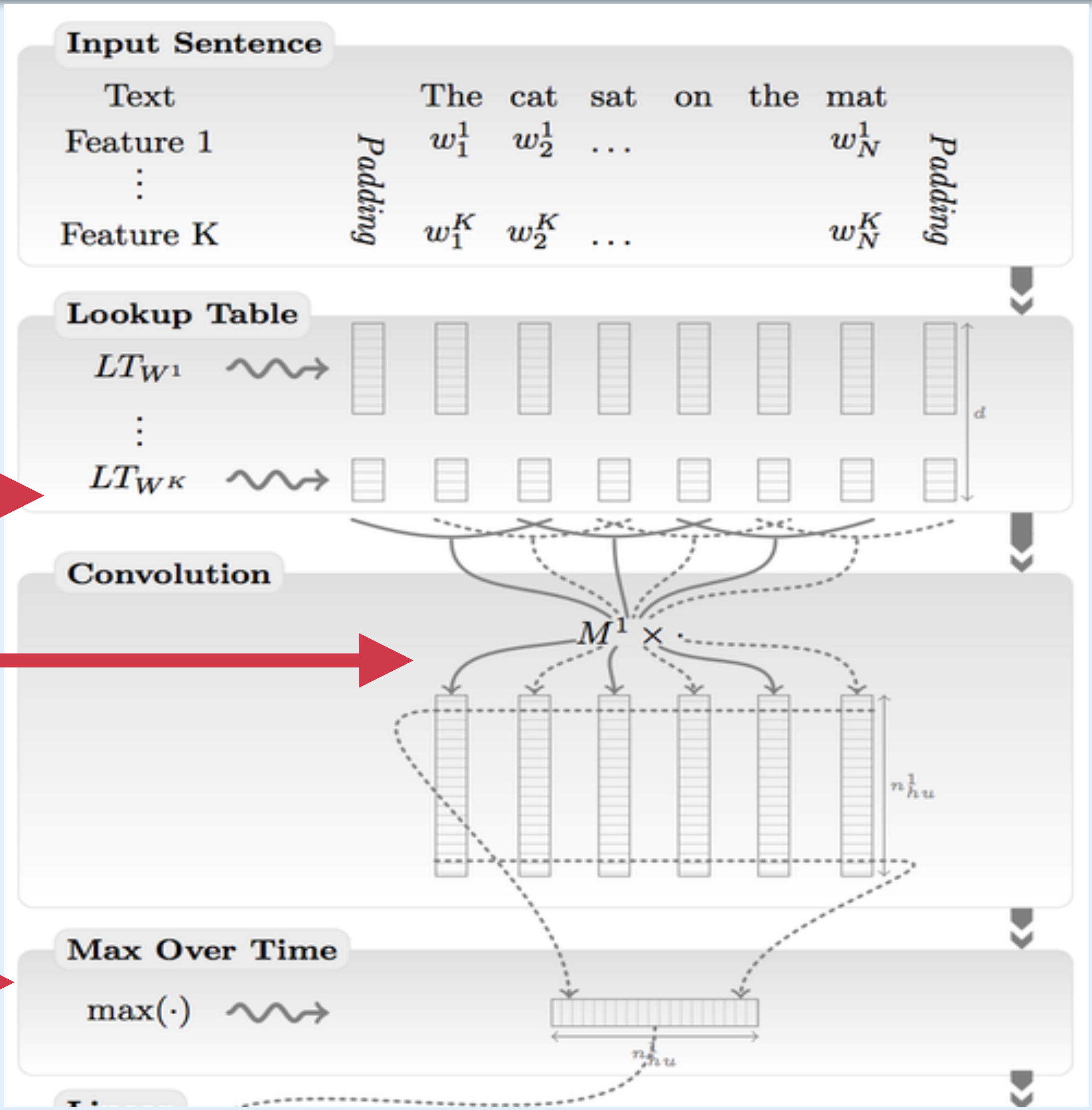
This model takes the whole sentence into consideration by means of overlapping windows.

CNN :: Architecture

Input Layer

Convolution Layer

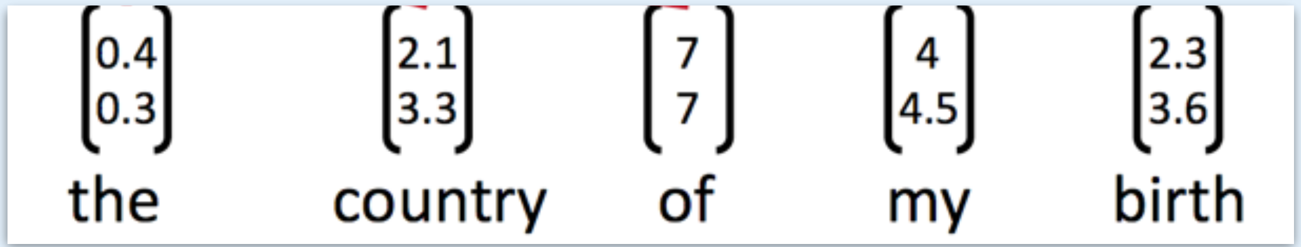
Max Pool Layer



CNN :: Layers (1/3)

Input Layer

- This comprises of the concatenated feature vectors of the words (**word embeddings**) in the input sentence.
- Word vectors with dimensionality 'k' (obtained from lookup table) :: $x_i \in \mathbb{R}^k$
- Sentence of length 'n' (input to the system) :: $x_{1:n} = x_1 \oplus x_2 \oplus \ldots \oplus x_n$
- Concatenation of words in range (i,j) :: $x_{i:i+j}$



$\begin{bmatrix}0.4\\0.3\end{bmatrix}$ the
$\begin{bmatrix}2.1\\3.3\end{bmatrix}$ country
$\begin{bmatrix}7\\7\end{bmatrix}$ of
$\begin{bmatrix}4\\4.5\end{bmatrix}$ my
$\begin{bmatrix}2.3\\3.6\end{bmatrix}$ birth

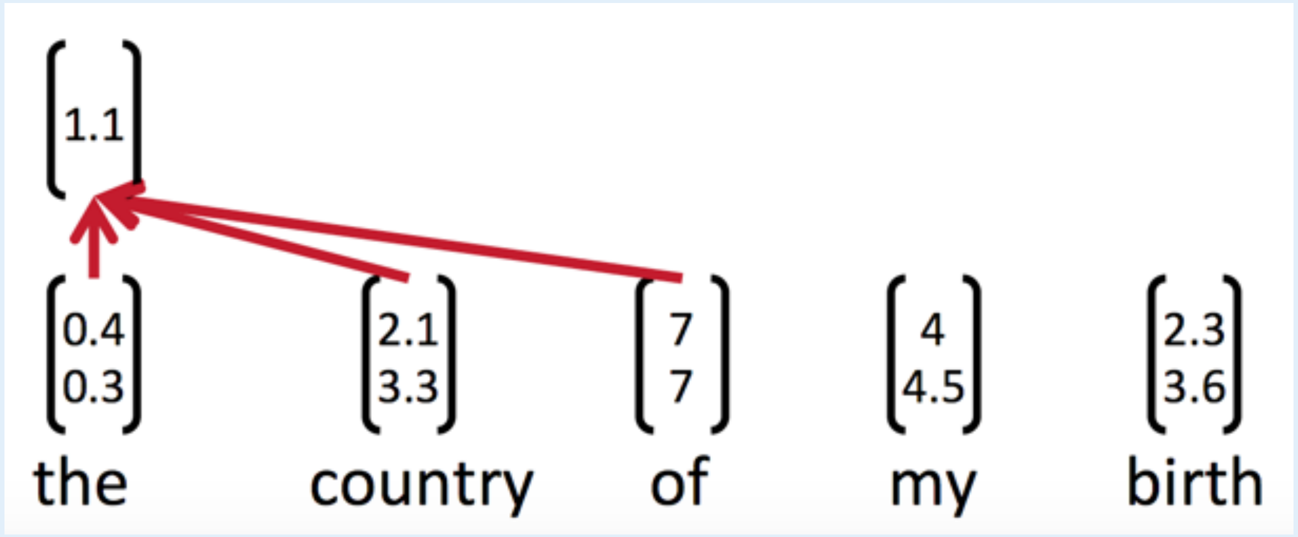
http://cs224d.stanford.edu/lectures/CS224d-Lecture13.pdf

CNN :: Layers (2/3)

Convolution Layer

- Generalisation of window approach where the linear layer operation is applied to successive overlapping windows of fixed size. (Padding is done accordingly).
- Convolutional filter :: $w \in \mathbb{R}^{hk}$

  — a vector which goes over a window of 'h' words
- Computation of a feature for CNN layer :: $c_i = f(w^T x_{i:i+h-1} + b)$

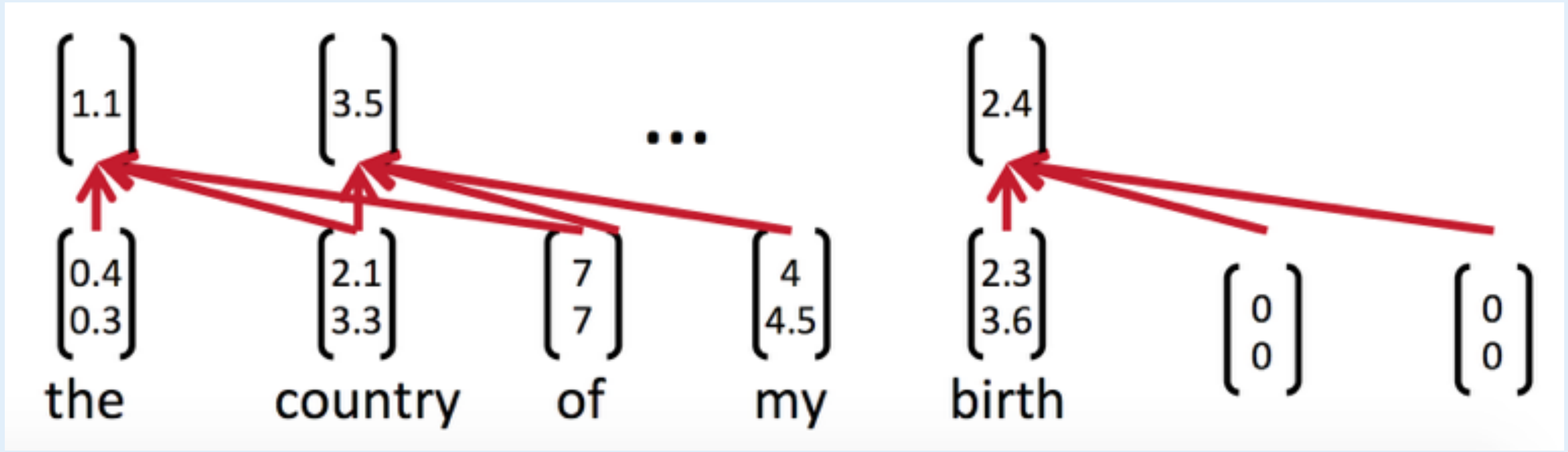


http://cs224d.stanford.edu/lectures/CS224d-Lecture13.pdf

CNN :: Layers (2/3)

Convolution Layer

- Filter '*w*' is applied to all possible windows (concatenated vectors)
  - — Padding is done accordingly
- Sentence of length '*n*' :: $x_{1:n} = x_1 \oplus x_2 \oplus \ldots \oplus x_n$
- All possible windows of length '*h*': $\{x_{1:h}, x_{2:h+1}, \ldots, x_{n-h+1:n}\}$
- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$



http://cs224d.stanford.edu/lectures/CS224d-Lecture13.pdf

CNN :: Layers (3/3)

Max Pooling Layer

- Max pooling operation is performed to select the most prominent features contributing to classification of sentence.

  — Averaging is not preferred for many NLP tasks because all words in a sentence do not contribute equally in tagging a word.

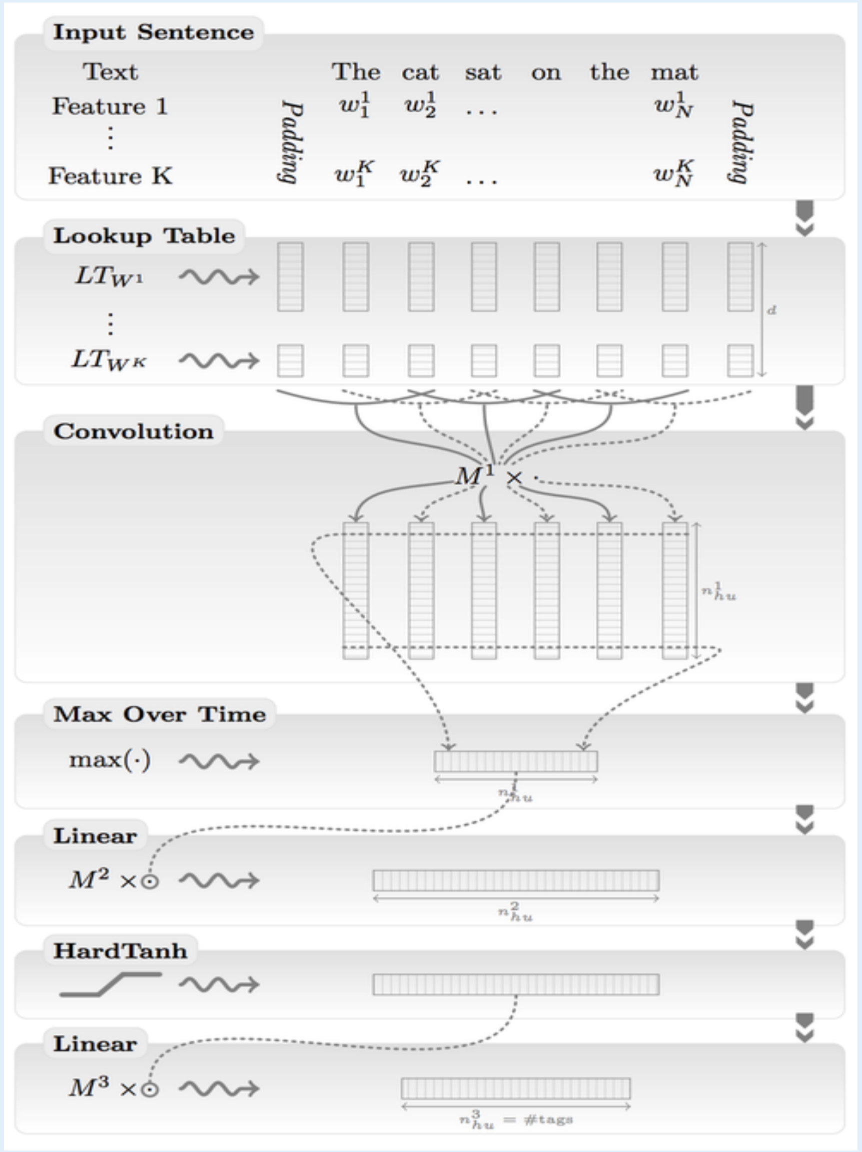
- Idea :: Capture the most important activation (maximum over time)
- Pooled single number :: $\hat{c} = max\{\mathbf{c}\}$

  — from feature map :: $\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}] \in \mathbb{R}^{n-h+1}$

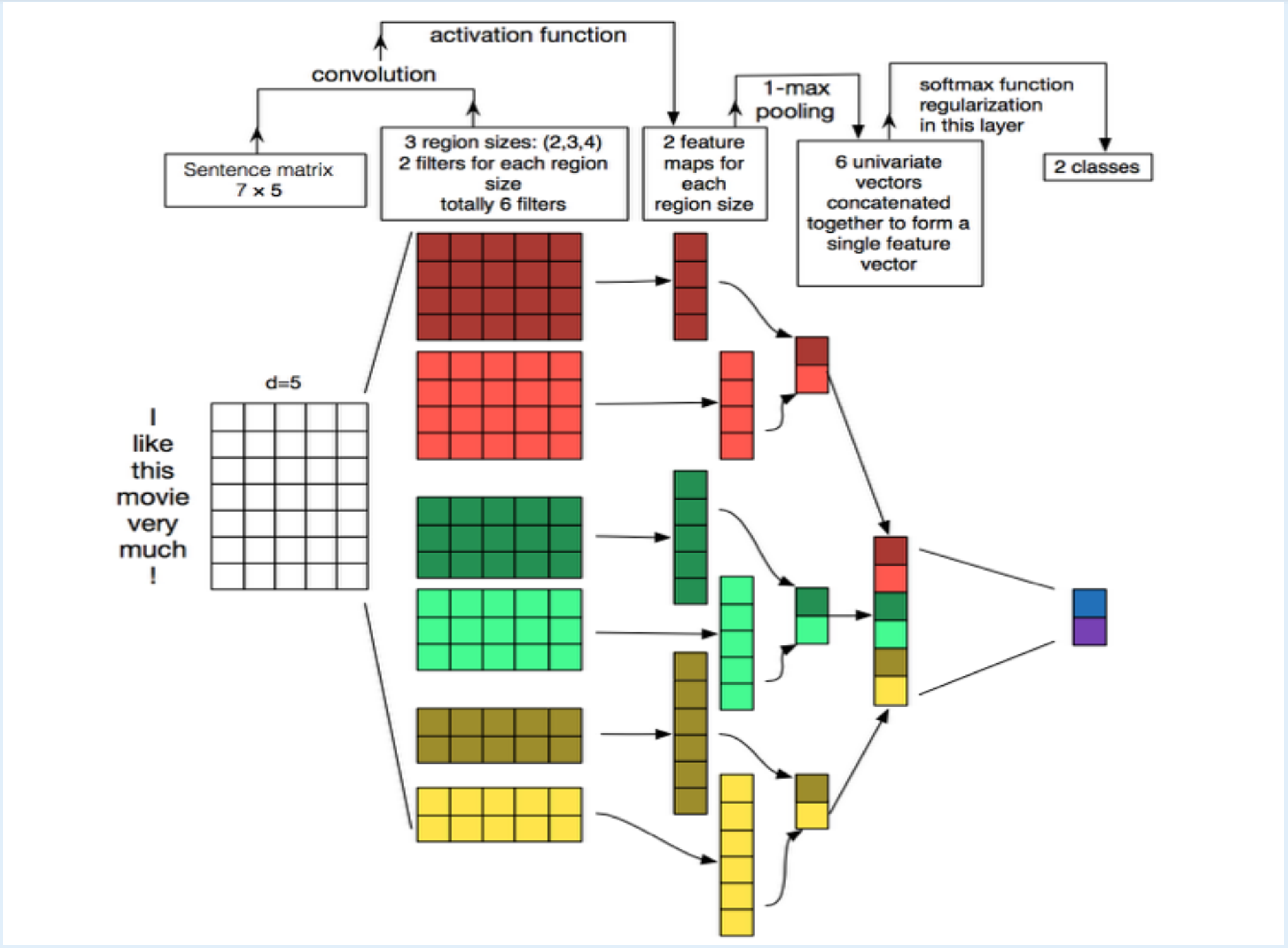
- **Because of max pooling, length of feature map (dependent on the length of the sentence) does not affect the architecture**

CNN :: Architecture

- Multiple features are required ::
  - — Use different window (filter) sizes (say unigrams, bigrams, trigrams, 4-grams, etc.)
  - — Use multiple weights for each filter
- The max-pool output is then passed through **fully connected neural network**
- As input, pre-trained word-vectors can be used (downloaded or trained using word2vec)
- Two versions ::
  - — **static ::** no change in word-embeddings
  - — **non-static ::** word-embeddings are also learned; updated through back-propagation



CNN :: Architecture for sentence classification


activation function
convolution
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
1-max pooling
6 univariate vectors concatenated together to form a single feature vector
softmax function regularization in this layer
2 classes
d=5
I
like
this
movie
very
much
!

Zhang, Ye, and Byron Wallace. "**A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification**." arXiv preprint arXiv:1510.03820 (2015).

CNN :: Summary

- CNN model looks very complicated, feels very advanced in architecture, and seems to be very effective to solve problems
- In fact, **convolutional neural networks use relatively little pre-processing. This means that the network is responsible for learning the filters that in traditional algorithms were hand-engineered. The lack of a dependence on prior-knowledge and the existence of difficult to design hand-engineered features is a major advantage for CNNs.** (wikipedia)
- Demands :: enough training data, enough training time
- Promises :: Give a chance to all inputs, select the informative and the essential ones and try to give as accurate a prediction for the problem at hand as possible

CNN :: References

- Y. Kim, “**Convolutional neural networks for sentence classification**,” *arXiv preprint arXiv:1408.5882*, 2014.
- R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “**Natural language processing (almost) from scratch**,” *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.
- Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. Acl, 655–665.
- Zhang, Ye, and Byron Wallace. "**A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification**." arXiv preprint arXiv:1510.03820 (2015).
- Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, “**A neural probabilistic language model**,” *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.
- R. M. Richard Socher, Milad Mohammadi, “*Deep learning for nlp*.” http://cs224d.stanford.edu/lecture_notes/LectureNotes13.pdf
- WildML, Understanding Convolutional Neural Networks for NLP, http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/
- Deep Learning - Convolutional Neural Networks, http://www.slideshare.net/perone/deep-learning-convolutional-neural-networks

Distributed Representations for Words

Kevin Patel

IIT Bombay

Outline

Recap

Its all about how you present your data

Its all about how you present your data

- The old adage: Garbage In Garbage Out

Its all about how you present your data

- The old adage: Garbage In Garbage Out
- Feature Engineering: Manually extracting and selecting features of data for learning

Its all about how you present your data

- The old adage: Garbage In Garbage Out
- Feature Engineering: Manually extracting and selecting features of data for learning
- Representation Learning: Doing this automatically

Representation Learning

- Learning representation of data that is best suited to the current task

Representation Learning

- Learning representation of data that is best suited to the current task
    - A *good* representation should answer the following questions positively:

Representation Learning

- Learning representation of data that is best suited to the current task
  - A *good* representation should answer the following questions positively:
    - What prior assumptions are being made?

Representation Learning

- Learning representation of data that is best suited to the current task
  - A *good* representation should answer the following questions positively:
    - What prior assumptions are being made?
    - Is it distributed?

Local Representations

- Information about a particular feature located solely in the corresponding dimension

Word	**Is living?**	**Is singular?**	**POS**
boy	Yes	Yes	Noun
eats	No	No	Verb

Distributed Representations

- Information about a particular feature distributed among a set of (not necessarily mutually exclusive) dimensions

Distributed Representations

- Information about a particular feature distributed among a set of (not necessarily mutually exclusive) dimensions
  - One feature spread over multiple dimensions
  - One dimension contributing to multiple features

Distributed Representations: Example

Number	Local Representation	Distributed Representation
0	1 0 0 0 0 0 0 0	0 0 0
1	0 1 0 0 0 0 0 0	0 0 1
2	0 0 1 0 0 0 0 0	0 1 0
3	0 0 0 1 0 0 0 0	0 1 1
4	0 0 0 0 1 0 0 0	1 0 0
5	0 0 0 0 0 1 0 0	1 0 1
6	0 0 0 0 0 0 1 0	1 1 0
7	0 0 0 0 0 0 0 1	1 1 1

Word Representations

- Words treated as atomic symbols

Word Representations

- Words treated as atomic symbols
- *Water, water, everywhere, not any drop to drink*
  - Example: In classical n-gram language modelling, $P(hotel|book, a)$ does not contribute at all to $P(motel|book, a)$

Word Representations

- Words treated as atomic symbols
- *Water, water, everywhere, not any drop to drink*
  - Example: In classical n-gram language modelling, $P(hotel|book, a)$ does not contribute at all to $P(motel|book, a)$
- Would be better if we can leverage knowledge of *hotel* while talking about *motel*, since they have similar meaning

Meaning of a word

- Applications that may benefit if meaning understood

Meaning of a word

- Applications that may benefit if meaning understood
    - Machine Translation

Meaning of a word

- Applications that may benefit if meaning understood
  - Machine Translation
  - Information Retrieval

Meaning of a word

- Applications that may benefit if meaning understood
  - Machine Translation
  - Information Retrieval
  - ...

Meaning of a word

- Applications that may benefit if meaning understood
  - Machine Translation
  - Information Retrieval
  - . . .
- What is the meaning of meaning?

Meaning of *Meaning* or What does *meaning* mean?

Meaning of *Meaning* or What does *meaning* mean?

- **Oxford Dictionary**: 'What is meant by a word, text, concept, or action'

Meaning of *Meaning* or What does *meaning* mean?

- **Oxford Dictionary**: 'What is meant by a word, text, concept, or action'
- **Princeton WordNet**: 'The message that is intended or expressed or signified'

Meaning of *Meaning* or What does *meaning* mean?

- **Oxford Dictionary**: 'What is meant by a word, text, concept, or action'
- **Princeton WordNet**: 'The message that is intended or expressed or signified'
- **Urban Dictionary**: 'What people try to create or find. A human condition in which they cannot exist in a meaningless state, even if they do live in a meaningless state, they need to pretend they exist in a world of meaning.'

Modelling meaning for NLP

Modelling meaning for NLP

- Do we need to understand meaning to the extent of its involvement in neurophysiologically correct mechanism of human information processing?

Modelling meaning for NLP

- Do we need to understand meaning to the extent of its involvement in neurophysiologically correct mechanism of human information processing?
  - We are not quite there yet.

Modelling meaning for NLP

- Do we need to understand meaning to the extent of its involvement in neurophysiologically correct mechanism of human information processing?
  - We are not quite there yet.
- Can we instead, have a computational model that is consistent with human behaviour?

Modelling meaning for NLP

- Do we need to understand meaning to the extent of its involvement in neurophysiologically correct mechanism of human information processing?
  - We are not quite there yet.
- Can we instead, have a computational model that is consistent with human behaviour?
  - Seems relatively feasible

Modelling meaning for NLP

- Do we need to understand meaning to the extent of its involvement in neurophysiologically correct mechanism of human information processing?
  - We are not quite there yet.
- Can we instead, have a computational model that is consistent with human behaviour?
  - Seems relatively feasible
  - Distributed Representations of words are examples of such models

Distributed Representations of words

Distributed Representations of words

- Also known as word vectors, word embeddings, etc.

Distributed Representations of words

- Also known as word vectors, word embeddings, etc.
- Primarily, they are vectors in n-dimensional space

Distributed Representations of words

- Also known as word vectors, word embeddings, etc.
- Primarily, they are vectors in n-dimensional space
- Try to model meaning of word

Distributed Representations of words

- Also known as word vectors, word embeddings, etc.
- Primarily, they are vectors in n-dimensional space
- Try to model meaning of word
- Salient points

Distributed Representations of words

- Also known as word vectors, word embeddings, etc.
- Primarily, they are vectors in n-dimensional space
- Try to model meaning of word
- Salient points
  - Based entirely on language data

Distributed Representations of words

- Also known as word vectors, word embeddings, etc.
- Primarily, they are vectors in n-dimensional space
- Try to model meaning of word
- Salient points
  - Based entirely on language data
    - Meaning of new word can also be acquired just through reading (Miller and Charles, 1991)

Distributed Representations of words

- Also known as word vectors, word embeddings, etc.
- Primarily, they are vectors in n-dimensional space
- Try to model meaning of word
- Salient points
  - Based entirely on language data
    - Meaning of new word can also be acquired just through reading (Miller and Charles, 1991)
  - No prior assumptions about language

Distributed Representations of words (contd.)

- Questions one should ask:

Distributed Representations of words (contd.)

- Questions one should ask:
  - Why are they *vectors*?

Distributed Representations of words (contd.)

- Questions one should ask:
  - Why are they *vectors*?
  - How to create them?

Distributed Representations of words (contd.)

- Questions one should ask:
  - Why are they *vectors*?
  - How to create them?
  - Are they a complete model of meaning?

    Or do they convey only specific aspects?

Distributed Representations of words (contd.)

- Questions one should ask:
  - Why are they *vectors*?
  - How to create them?
  - Are they a complete model of meaning?

    Or do they convey only specific aspects?
  - Is it possible to extract meaning by merely looking at usage data?

Why are they *vectors*?

Why are they *vectors*?

- Thanks to psychology (Lakoff and Johnson, 1980, 1999)

Why are they *vectors*?

- Thanks to psychology (Lakoff and Johnson, 1980, 1999)
- **Similarity-is-Proximity**: two *similar* things are conceptualized as being *close to* or *near* each other

Why are they *vectors*?

- Thanks to psychology (Lakoff and Johnson, 1980, 1999)
- **Similarity-is-Proximity**: two *similar* things are conceptualized as being *close to* or *near* each other
- **Entities-are-Locations**: in order for two things to be *close to* each other, they need to have a spatial location

Why are they *vectors*?

- Thanks to psychology (Lakoff and Johnson, 1980, 1999)
- **Similarity-is-Proximity**: two *similar* things are conceptualized as being *close to* or *near* each other
- **Entities-are-Locations**: in order for two things to be *close to* each other, they need to have a spatial location
- **Geometric Metaphor of meaning**: Meanings are points in space, and the proximity among their locations is a measure of their semantic similarity (Sahlgren, 2006)

Entire Vector vs. Individual dimensions

Entire Vector vs. Individual dimensions

- Only proximity in the entire space is represented

Entire Vector vs. Individual dimensions

- Only proximity in the entire space is represented
- No phenomenological correlations with dimensions of high-dimensional space (in majority of algorithms)

Entire Vector vs. Individual dimensions

- Only proximity in the entire space is represented
- No phenomenological correlations with dimensions of high-dimensional space (in majority of algorithms)
    - Those models who do have some correlations, are known as *interpretable models*

How to create them?

- Consider the following sentences:
  1. Can you cook some *xyzerfw* for me?
  2. This *xyzerfw* are so delicious.
  3. *xyzerfw* are not as healthy as fresh vegetables.

How to create them?

- Consider the following sentences:
  1. Can you cook some *xyzerfw* for me?
  2. This *xyzerfw* are so delicious.
  3. *xyzerfw* are not as healthy as fresh vegetables.
- Can you guess the meaning of xyzerfw ?

How to create them?

- Consider the following sentences:
  1. Can you cook some *xyzerfw* for me?
  2. This *xyzerfw* are so delicious.
  3. *xyzerfw* are not as healthy as fresh vegetables.
- Can you guess the meaning of xyzerfw ?
- How about now?
  4. Maggi *xyzerfw* were recently banned for a brief period of time.

How to create them? (contd.)

- **Harris Distributional Hypothesis**: Words with similar distributional properties have similar meanings. (Harris, 1970)

How to create them? (contd.)

- **Harris Distributional Hypothesis**: Words with similar distributional properties have similar meanings. (Harris, 1970)
- Harris does mentions that distributional approaches can model differences in meaning rather than the proper meaning itself

How to create them? (contd.)

- *Semantic differential approaches* to meaning representations
  - Example: (Osgood, 1952)

	small-large	bald-furry	docile-dangerous
mouse	2	6	1
rat	2	6	4

How to create them? (contd.)

- *Semantic differential approaches* to meaning representations
  - Example: (Osgood, 1952)

	small-large	bald-furry	docile-dangerous
mouse	2	6	1
rat	2	6	4

- Major problems:

How to create them? (contd.)

- *Semantic differential approaches* to meaning representations
  - Example: (Osgood, 1952)

	small-large	bald-furry	docile-dangerous
mouse	2	6	1
rat	2	6	4

- Major problems:
  - Features defined manually

How to create them? (contd.)

- *Semantic differential approaches* to meaning representations
  - Example: (Osgood, 1952)

	small-large	bald-furry	docile-dangerous
mouse	2	6	1
rat	2	6	4

- Major problems:
  - Features defined manually
  - Allowed limited number of semantic features

How to create them? (contd.)

- *Semantic differential approaches* to meaning representations
  - Example: (Osgood, 1952)

	small-large	bald-furry	docile-dangerous
mouse	2	6	1
rat	2	6	4

- Major problems:
  - Features defined manually
  - Allowed limited number of semantic features
  - Is it theoretically possible to come up with limited set of features to exhaustively cover the meaning space?

Dynamic Context Vectors

Dynamic Context Vectors

- A really cool answer to 'How to create them' by Gallant (1991)

Dynamic Context Vectors

- A really cool answer to 'How to create them' by Gallant (1991)
- Fixed dimensions of existing semantic vectors plus additional dimensions
    - Additional dimensions initialized randomly
    - Modified during learning such that proximity with vectors of neighbours increases

Co-occurrence Matrix

Co-occurrence Matrix

- Originally proposed by Schütze (1992)

Co-occurrence Matrix

- Originally proposed by Schütze (1992)
- Foundation of count based approaches that follow

Co-occurrence Matrix

- Originally proposed by Schütze (1992)
- Foundation of count based approaches that follow
- Automatic derivation of vectors

Co-occurrence Matrix

- Originally proposed by Schütze (1992)
- Foundation of count based approaches that follow
- Automatic derivation of vectors
- Collect co-occurrence counts in a matrix

Co-occurrence Matrix

- Originally proposed by Schütze (1992)
- Foundation of count based approaches that follow
- Automatic derivation of vectors
- Collect co-occurrence counts in a matrix
- Rows or columns are the vectors of corresponding word

Co-occurrence Matrix

- Originally proposed by Schütze (1992)
- Foundation of count based approaches that follow
- Automatic derivation of vectors
- Collect co-occurrence counts in a matrix
- Rows or columns are the vectors of corresponding word
- If counting in both directions, matrix is symmetrical
- If counting in one side, matrix is asymmetrical, and is known as directional co-occurrence

Co-occurrence Matrix: Example

- Toy Corpus
  - I hate rough driving .
  - I hate databases .
  - I enjoy flying .

Co-occurrence matrix: Example (contd.)

counts	I	hate	enjoy	rough	driving	databases	flying	.
I	0	2	1	0	0	0	0	0
hate	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
rough	0	1	0	0	1	0	0	0
driving	0	0	0	1	0	0	0	1
databases	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Table: Window 1 Symmetric Co-occurrence matrix for the sample corpus

Similarity between vectors

- Simple alternatives such as dot product, Minkowski metrics, etc. exist
- Cosine Similarity is preferred

Similarity between vectors

- Simple alternatives such as dot product, Minkowski metrics, etc. exist
- Cosine Similarity is preferred
    - Fixed range of similarity

Similarity between vectors

- Simple alternatives such as dot product, Minkowski metrics, etc. exist
- Cosine Similarity is preferred
    - Fixed range of similarity
    - Considers normalized vectors

Similarity between vectors

- Simple alternatives such as dot product, Minkowski metrics, etc. exist
- Cosine Similarity is preferred
    - Fixed range of similarity
    - Considers normalized vectors
    - Efficient to compute

Count based approaches

- Uses Co-occurrence counts

Count based approaches

- Uses Co-occurrence counts
- We shall look at
  - Latent Semantic Analysis (Dumais et al., 1988)
  - Hyperspace Analogous to Language (Lund and Burgess, 1996)
  - Random Indexing (Sahlgren, 2005)

LSA

- Latent Semantic Analysis

LSA

- Latent Semantic Analysis
- Originally developed as Latent Semantic Indexing (LSI) (Dumais et al., 1988)

LSA

- Latent Semantic Analysis
- Originally developed as Latent Semantic Indexing (LSI) (Dumais et al., 1988)
- Adapted for word-space models

LSA

- Latent Semantic Analysis
- Originally developed as Latent Semantic Indexing (LSI) (Dumais et al., 1988)
- Adapted for word-space models
- Developed to tackle inability of models of co-occurrence matrices to handle synonymy
    - Query about *hotels* cannot retrieve results about *motels*

LSA

- Latent Semantic Analysis
- Originally developed as Latent Semantic Indexing (LSI) (Dumais et al., 1988)
- Adapted for word-space models
- Developed to tackle inability of models of co-occurrence matrices to handle synonymy
  - Query about *hotels* cannot retrieve results about *motels*
- Words and Documents dimensions $\rightarrow$ Latent dimensions
  - Uses Singular Value Decomposition (SVD) for dimensionality reduction

LSA (contd.)

LSA (contd.)

- Words-by-documents matrix

LSA (contd.)

- Words-by-documents matrix
- Entropy based weighting of co-occurrences

$$f_{ij} = (log(TF_{ij}) + 1) \times (1 - (\sum_j (\frac{p_{ij} logp_{ij}}{logD}))) \tag{1}$$

where D is number of documents, $p_{ij} = \frac{TF_{ij}}{f_i}$, and f_i is frequency of term i in document collection

LSA (contd.)

- Words-by-documents matrix
- Entropy based weighting of co-occurrences

$$f_{ij} = (log(TF_{ij}) + 1) \times (1 - (\sum_j (\frac{p_{ij} log p_{ij}}{log D}))) \tag{1}$$

where D is number of documents, $p_{ij} = \frac{TF_{ij}}{f_i}$, and f_i is frequency of term i in document collection

- Truncated SVD to reduce dimensionality

LSA (contd.)

- Words-by-documents matrix
- Entropy based weighting of co-occurrences

$$f_{ij} = (log(TF_{ij}) + 1) \times (1 - (\sum_j (\frac{p_{ij} log p_{ij}}{log D}))) \quad (1)$$

  where D is number of documents, $p_{ij} = \frac{TF_{ij}}{f_i}$, and f_i is frequency of term i in document collection
- Truncated SVD to reduce dimensionality
- Cosine measure to compute vector similarities

HAL

- Hyperspace Analogous to Language (Lund and Burgess, 1996)

HAL

- Hyperspace Analogous to Language (Lund and Burgess, 1996)
- Developed specifically for word representations

HAL

- Hyperspace Analogous to Language (Lund and Burgess, 1996)
- Developed specifically for word representations
- Uses directional co-occurrence

HAL (contd.)

HAL (contd.)

- Directional word by word matrix

HAL (contd.)

- Directional word by word matrix
- Distance weighting of the co-occurrences

HAL (contd.)

- Directional word by word matrix
- Distance weighting of the co-occurrences
- Concatenation of row-column vectors

HAL (contd.)

- Directional word by word matrix
- Distance weighting of the co-occurrences
- Concatenation of row-column vectors
- Dimensionality reduction Optional
    - Discard low variant dimensions

HAL (contd.)

- Directional word by word matrix
- Distance weighting of the co-occurrences
- Concatenation of row-column vectors
- Dimensionality reduction Optional
  - Discard low variant dimensions
- Normalization of vectors to unit length

HAL (contd.)

- Directional word by word matrix
- Distance weighting of the co-occurrences
- Concatenation of row-column vectors
- Dimensionality reduction Optional
  - Discard low variant dimensions
- Normalization of vectors to unit length
- Similarities computed through Minkowski metric

RI

RI

- Random Indexing (Sahlgren, 2005)

RI

- Random Indexing (Sahlgren, 2005)
- Designed to tackle dimensionality from scratch

RI (contd.)

RI (contd.)

- Associate with each word a random vector

RI (contd.)

- Associate with each word a random vector
- Every time some words co-occur, add their vectors

RI (contd.)

- Associate with each word a random vector
- Every time some words co-occur, add their vectors
- Average and normalize the vectors

Prediction based approaches

- Try to predict either

Prediction based approaches

- Try to predict either
  - Word given context, or

Prediction based approaches

- Try to predict either
  - Word given context, or
  - Context given word

Prediction based approaches

- Try to predict either
  - Word given context, or
  - Context given word
- We shall look at
  - Neural Network Language Model (NNLM) (Bengio et al., 2003)
  - Skip Grams (Mikolov et al., 2013b,a)
  - GloVe (Pennington et al., 2014)

NNLM

- Neural Network Language Model

NNLM

- Neural Network Language Model
- Proposed by Bengio et al. (2003)

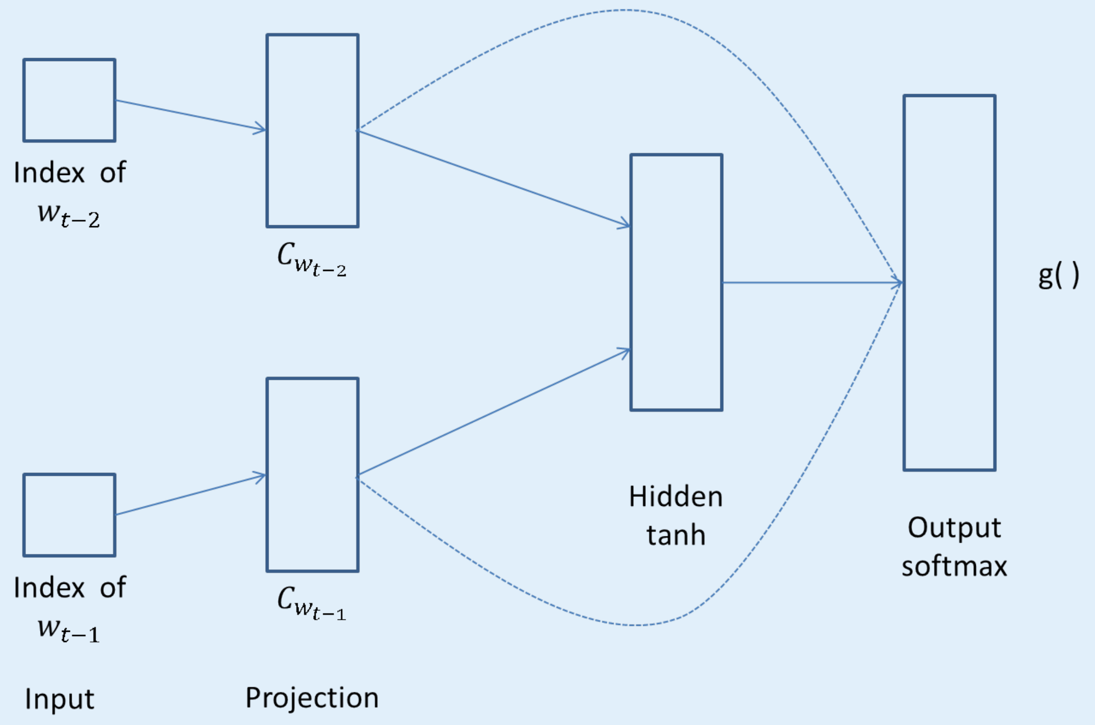
NNLM

- Neural Network Language Model
- Proposed by Bengio et al. (2003)
- Predict word given context

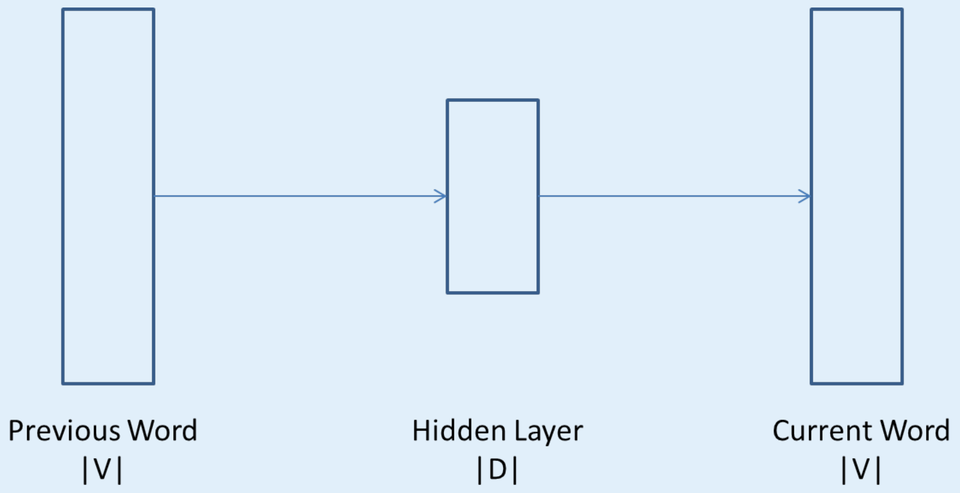
NNLM

- Neural Network Language Model
- Proposed by Bengio et al. (2003)
- Predict word given context
- Word Vectors learnt as a by-product of language modelling

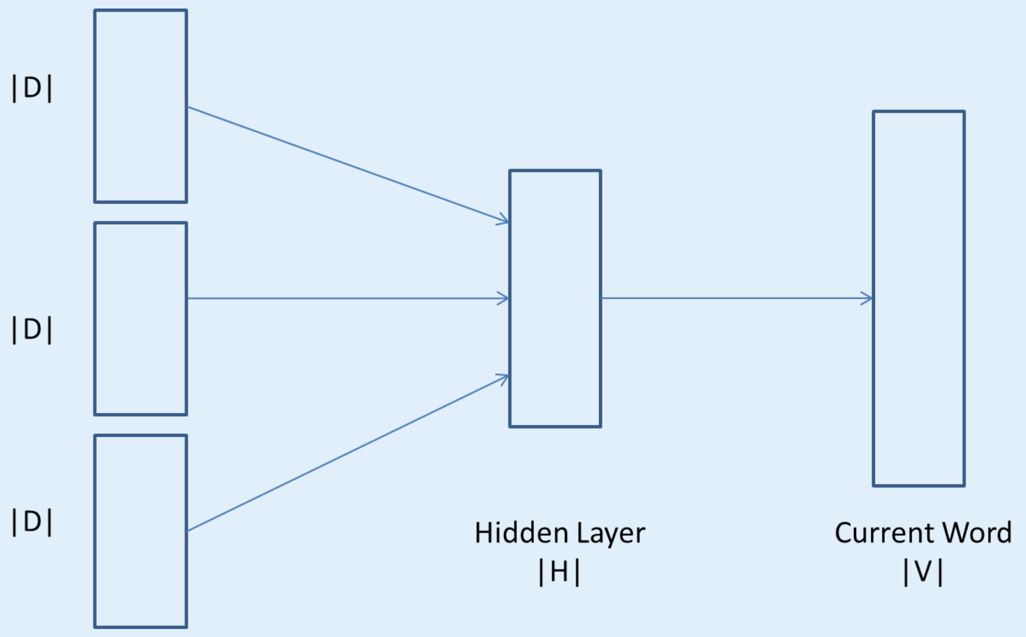
NNLM: Original Model



NNLM: Simplified (1)



NNLM: Simplified (2)



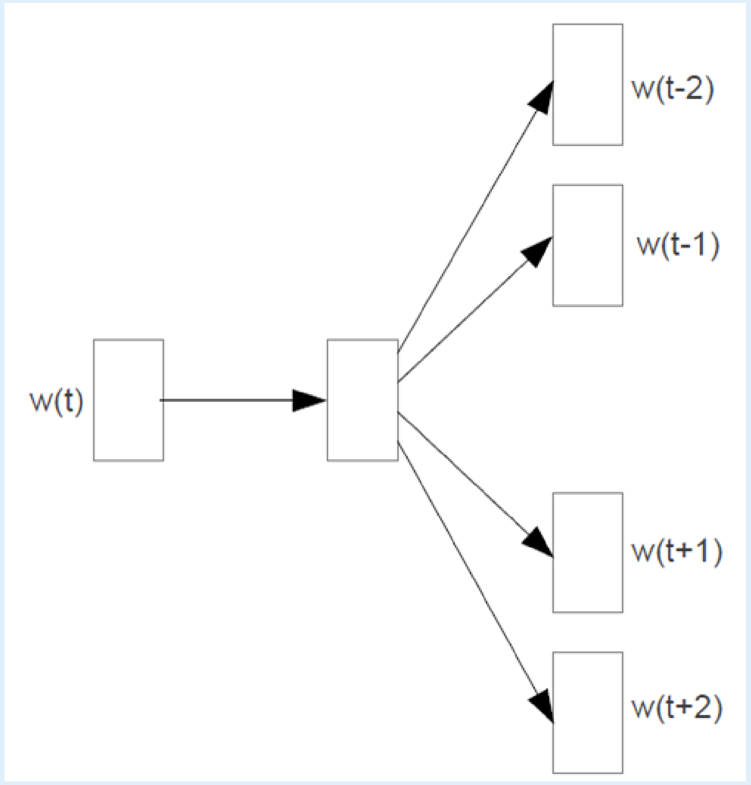
Skip Gram

Skip Gram

- Proposed by Mikolov et al. (2013b)

Skip Gram

- Proposed by Mikolov et al. (2013b)
- Predict Context given word



Skip Gram (contd.)

- Given a sequence of training words $w_1, w_2, \ldots, w_T$, maximize

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c,j}\log p\left(w_{t+j}|w_t\right) \tag{2}$$

where

$$p(w_O|w_I) = \frac{\exp(u_{w_O}^T v_{w_I})}{\sum_{w=1}^{W}\exp(u_w^T v_{w_I})} \tag{3}$$

GloVe

- Global Vectors

GloVe

- Global Vectors
- Proposed by Pennington et al. (2014)

GloVe

- Global Vectors
- Proposed by Pennington et al. (2014)
- Predict Context given word

GloVe

- Global Vectors
- Proposed by Pennington et al. (2014)
- Predict Context given word
- Similar to Skip-gram, but objective function is different

$$J = \sum_{i,j=1}^{V} f(X_{ij})({w_i}^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2 \tag{4}$$

where X_{ij} can be likelihood of i_{th} and j^{th} word occuring together, and f is a weightage function

How to evaluate your word vectors?

- Types of Evaluation

How to evaluate your word vectors?

- Types of Evaluation
  - Extrinsic: Use embeddings as features in downstream tasks such as POS, NER, etc.

How to evaluate your word vectors?

- Types of Evaluation
  - Extrinsic: Use embeddings as features in downstream tasks such as POS, NER, etc.
  - Intrinsic: Evaluate linguistic regularities exhibited by word vectors

How to evaluate your word vectors?

- Types of Evaluation
  - Extrinsic: Use embeddings as features in downstream tasks such as POS, NER, etc.
  - Intrinsic: Evaluate linguistic regularities exhibited by word vectors
- Different embeddings' developers evaluate embeddings differently

How to evaluate your word vectors?

- Types of Evaluation
  - Extrinsic: Use embeddings as features in downstream tasks such as POS, NER, etc.
  - Intrinsic: Evaluate linguistic regularities exhibited by word vectors
- Different embeddings' developers evaluate embeddings differently
- Some groups have done evaluations on standard sets

How to evaluate your word vectors?

- Types of Evaluation
  - Extrinsic: Use embeddings as features in downstream tasks such as POS, NER, etc.
  - Intrinsic: Evaluate linguistic regularities exhibited by word vectors
- Different embeddings' developers evaluate embeddings differently
- Some groups have done evaluations on standard sets
- Will discuss some of the intrinsic evaluation mechanisms

Word Pair Similarity task

Word Pair Similarity task

- Given a pair of words (w_1, w_2), find similarity

Word Pair Similarity task

- Given a pair of words (w_1, w_2), find similarity
- Different datasets exist for such evaluation

Word Pair Similarity task

- Given a pair of words (w_1, w_2), find similarity
- Different datasets exist for such evaluation
- `http://www.wordvectors.org` - A common web platform with multiple datasets (Faruqui and Dyer, 2014)

Word Pair Similarity task

- Given a pair of words (w_1, w_2), find similarity
- Different datasets exist for such evaluation
- `http://www.wordvectors.org` - A common web platform with multiple datasets (Faruqui and Dyer, 2014)
    - 12 different datasets

Word Pair Similarity task

- Given a pair of words (w_1, w_2), find similarity
- Different datasets exist for such evaluation
- `http://www.wordvectors.org` - A common web platform with multiple datasets (Faruqui and Dyer, 2014)
  - 12 different datasets
  - 7 different pre-trained word vectors available to compare with

Word Pair Similarity task

- Given a pair of words (w_1, w_2), find similarity
- Different datasets exist for such evaluation
- `http://www.wordvectors.org` - A common web platform with multiple datasets (Faruqui and Dyer, 2014)
  - 12 different datasets
  - 7 different pre-trained word vectors available to compare with
  - Provides t-SNE visualizations for antonym-synonym and male-female

Word Analogy task

Word Analogy task

- Proposed by Mikolov et al. (2013b)

Word Analogy task

- Proposed by Mikolov et al. (2013b)
- Try to answer the question

  man is to woman as king is to ?

Word Analogy task

- Proposed by Mikolov et al. (2013b)
- Try to answer the question

  man is to woman as king is to ?
- Often discussed in media

Word Intrusion detection task

Word Intrusion detection task

- Proposed by Murphy et al. (2012)

Word Intrusion detection task

- Proposed by Murphy et al. (2012)
- Provides a way to interpret dimensions

Word Intrusion Detection task (contd.)

- The task:

Word Intrusion Detection task (contd.)

- The task:
  1. Select a dimension

Word Intrusion Detection task (contd.)

- The task:
  1. Select a dimension
  2. Reverse sort all vectors based on this dimension

Word Intrusion Detection task (contd.)

- The task:
  1. Select a dimension
  2. Reverse sort all vectors based on this dimension
  3. Select top 5 words

Word Intrusion Detection task (contd.)

- The task:
  1. Select a dimension
  2. Reverse sort all vectors based on this dimension
  3. Select top 5 words
  4. Select a word, which is in bottom half of this list, and is in top 10 percentile in some other columns

Word Intrusion Detection task (contd.)

- The task:
    1. Select a dimension
    2. Reverse sort all vectors based on this dimension
    3. Select top 5 words
    4. Select a word, which is in bottom half of this list, and is in top 10 percentile in some other columns
    5. Give a random permutation of these 6 words to a human evaluator
        - Example: {bathroom, closet, attic, balcony, quickly, hall}

Word Intrusion Detection task (contd.)

- The task:
  1. Select a dimension
  2. Reverse sort all vectors based on this dimension
  3. Select top 5 words
  4. Select a word, which is in bottom half of this list, and is in top 10 percentile in some other columns
  5. Give a random permutation of these 6 words to a human evaluator
     - Example: {bathroom, closet, attic, balcony, quickly, hall}
  6. Check precision

Word Intrusion detection task

Word Intrusion detection task

- Most approaches do not report results on this task

Word Intrusion detection task

- Most approaches do not report results on this task
  - Experiments done by us suggest many of them are not interpretable

What next?

- Vectors of complex entities

What next?

- Vectors of complex entities
  - Phrases
  - Sentences
  - Documents
  - Synsets

Summary

Summary

- Motivated distributed representations in general

Summary

- Motivated distributed representations in general
- Discussed word vectors in detail

Summary

- Motivated distributed representations in general
- Discussed word vectors in detail
  - Choice of vectors as mathematical structure for representing words

Summary

- Motivated distributed representations in general
- Discussed word vectors in detail
    - Choice of vectors as mathematical structure for representing words
    - Gathering information for creating vectors

Summary

- Motivated distributed representations in general
- Discussed word vectors in detail
    - Choice of vectors as mathematical structure for representing words
    - Gathering information for creating vectors
    - Discussed few word vector models

Summary

- Motivated distributed representations in general
- Discussed word vectors in detail
  - Choice of vectors as mathematical structure for representing words
  - Gathering information for creating vectors
  - Discussed few word vector models
  - Provided evaluation mechanisms

References I

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Dumais, S. T., Furnas, G. W., Landauer, T. K., Deerwester, S., and Harshman, R. (1988). Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285. ACM.

Faruqui, M. and Dyer, C. (2014). Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, USA. Association for Computational Linguistics.

References II

Gallant, S. I. (1991). A practical approach for representing context and for performing word sense disambiguation using neural networks. *Neural Computation*, 3(3):293–309.

Harris, Z. S. (1970). *Distributional structure*. Springer.

Lakoff, G. and Johnson, M. (1980). Metaphors we live by. *Chicago, IL: University of*.

Lakoff, G. and Johnson, M. (1999). Philosophy in the flesh.

Lund, K. and Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

References III

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Miller, G. A. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

Murphy, B., Talukdar, P., and Mitchell, T. (2012). *Learning Effective and Interpretable Semantic Models using Non-Negative Sparse Embedding*, pages 1933–1949. Association for Computational Linguistics.

Osgood, C. E. (1952). The nature and measurement of meaning. *Psychological bulletin*, 49(3):197.

References IV

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Sahlgren, M. (2005). An introduction to random indexing. In *Methods and applications of semantic indexing workshop at the 7th international conference on terminology and knowledge engineering, TKE*, volume 5.

Sahlgren, M. (2006). The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces.

Schütze, H. (1992). Dimensions of meaning. In *Supercomputing'92., Proceedings*, pages 787–796. IEEE.

Q & A

Part III :: DEMO / HANDS-ON

Theano Basics & Demo

Prerana Singhal

IIT Bombay

Python-based Tools

- **Python**

  — Object Oriented Language, and Intuitive to coding; close to natural language
- **NumPy**

  — n-dimensional array object, and scientific computing toolbox
- **SciPy**

  — more scientific toolboxes, and sparse matrix objects
- **libgpuarray**

  — n-dimensional array objects in C for CUDA and OpenCL
- **Theano**

  — Abstraction for machine learning; compiler/symbolic graph manipulation
- **Theanets**

  — Abstraction for neural networks, and optimized algorithms

Python

- General purpose high-level object oriented interpreted language
- Emphasises the code readability
- Comprehensive standard library
- Dynamic type and memory management
- Slow execution
- Easily extensible with C
- Popular in web development and scientific communities

NumPy/SciPy

- **NumPy**
  - *n*-dimensional numeric array for high-performance computing
  - Slice of array are views; no copy
  - Elementwise computations
  - Includes linear algebra and fourier transforms
  - Pseudo-random number generators
- **Scipy**
  - Sparse matrices
  - More linear algebra
  - Solvers and optimization algorithms
  - Matlab-compatible I/O
  - I/O and signal processing for image and audio

What is missing?

- Non-lazy evaluation hurts performance
- Bound to the CPU
- Lacks symbolic or automatic differentiation
- No automatic speed and stability optimisation

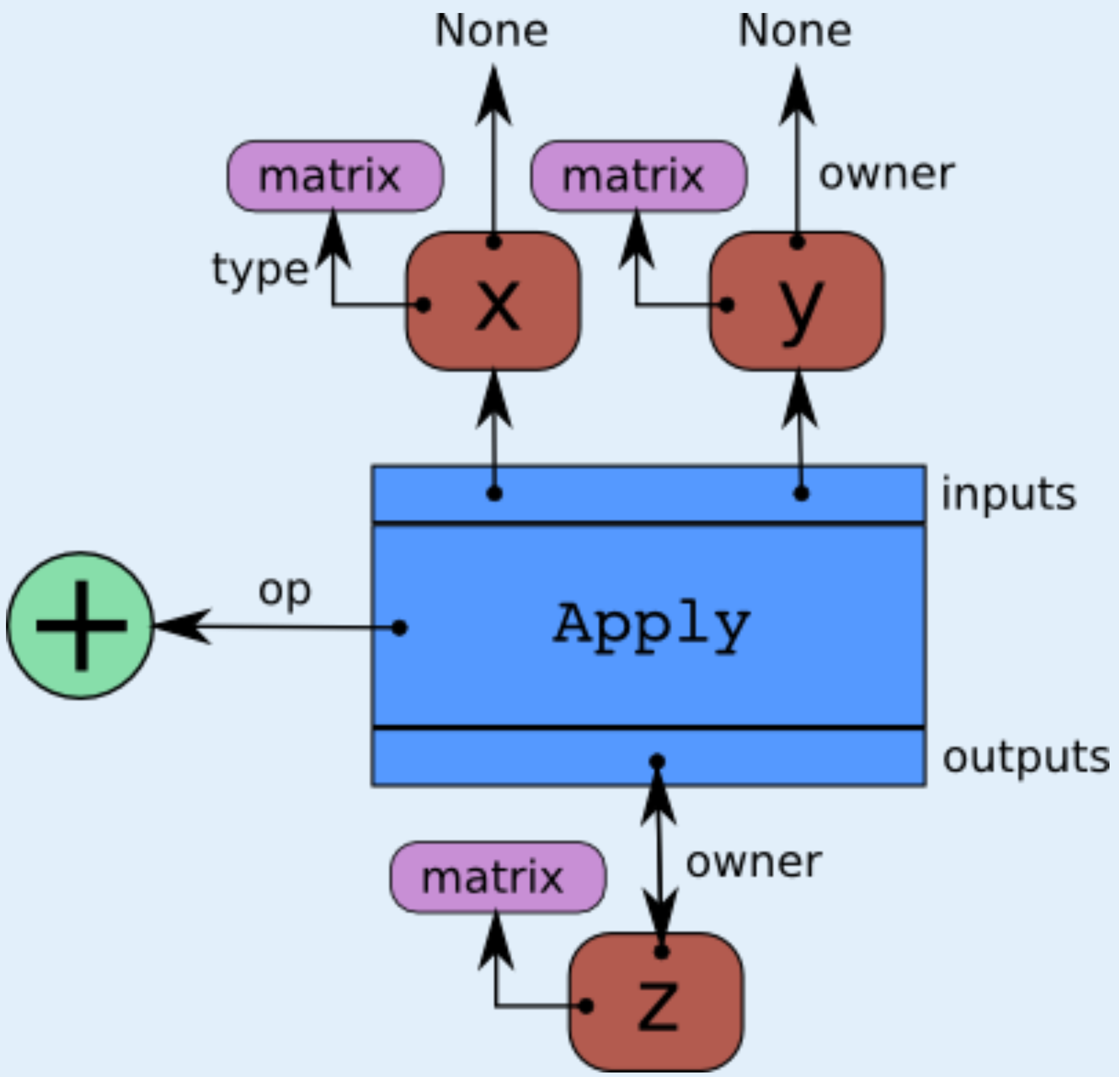
Theano

- Even higher level language; specially for machine learning related computation
- Syntax close to NumPy
- (most) compilation in C for CPU or GPU
- Automatic optimisation (speed and stability)
- Can reuse other library for best performance
  - BLAS, SciPy, Cython, Numba, PyCUDA, CUDA
- Automatic differentiation
- Sparse matrices

Simple Example

```
import theano

# declare symbolic variable
a = theano.tensor.vector(“a”)

# build symbolic expression
b = a + a ** 10

# compile function
f = theano.function([a], b)
print f([0, 1, 2])  # prints “array([0, 2, 1026])”
```

Computational Unit



Demo Outline

- Prerequisite (before a neural network)
  - — NumPy refresher
  - — Variables in theano
  - — Logistic function
  - — Shared Variables
- A simple neural network
  - — Feed-forward
  - — Back propagation

References

Theano Tutorial

- Theanets Hello world: simple classification example http://www.neural.cz/theanets-hello-world.html
- Ipython Notebook: http://nbviewer.ipython.org/github/craffel/theano-tutorial/blob/master/Theano%20Tutorial.ipynb
- Specific to Deep Learning: http://deeplearning.net/software/theano/tutorial/

Theano Documentation

- Stable version: http://deeplearning.net/software/theano/index.html
- Theanets 0.7.0pre release: http://theanets.readthedocs.org/en/latest/index.html

Credits :: Girishkumar Ponkiya, PhD, IIT Bombay

Demo: word2vec, sent2vec, doc2vec

Sudha Bhingardive

Indian Institute of Technology Bombay, Mumbai

word2vec demo

Outline

Introduction: word2vec

- A tool which provides an efficient implementation of neural network architectures for computing vector representations of words
  - Skip-Gram model
  - Continuous Bag of Words (CBOW) model
- Word vectors encode valuable semantic information about the words that they represent
- **Input**: an unlabeled corpus
- **Output**: vector representation for each word in the corpus

word2vec: Download and Compile

- **Download**: `http://word2vec.googlecode.com/svn/trunk/`
- **Compile**: make

word2vec: Training

- **Training**: find the script ./demo-word.sh in `word2vec` package
- `word2vec`
  `- train` <training_data>
  `- output` <file_name>
  `- window` <window_size>
  `- cbow` <0 (skip_gram), 1 (cbow)>
  `- size` <vector_size>
  `- binary` <0 (text), 1 (binary)>
  `- iter` <iteratio_num>

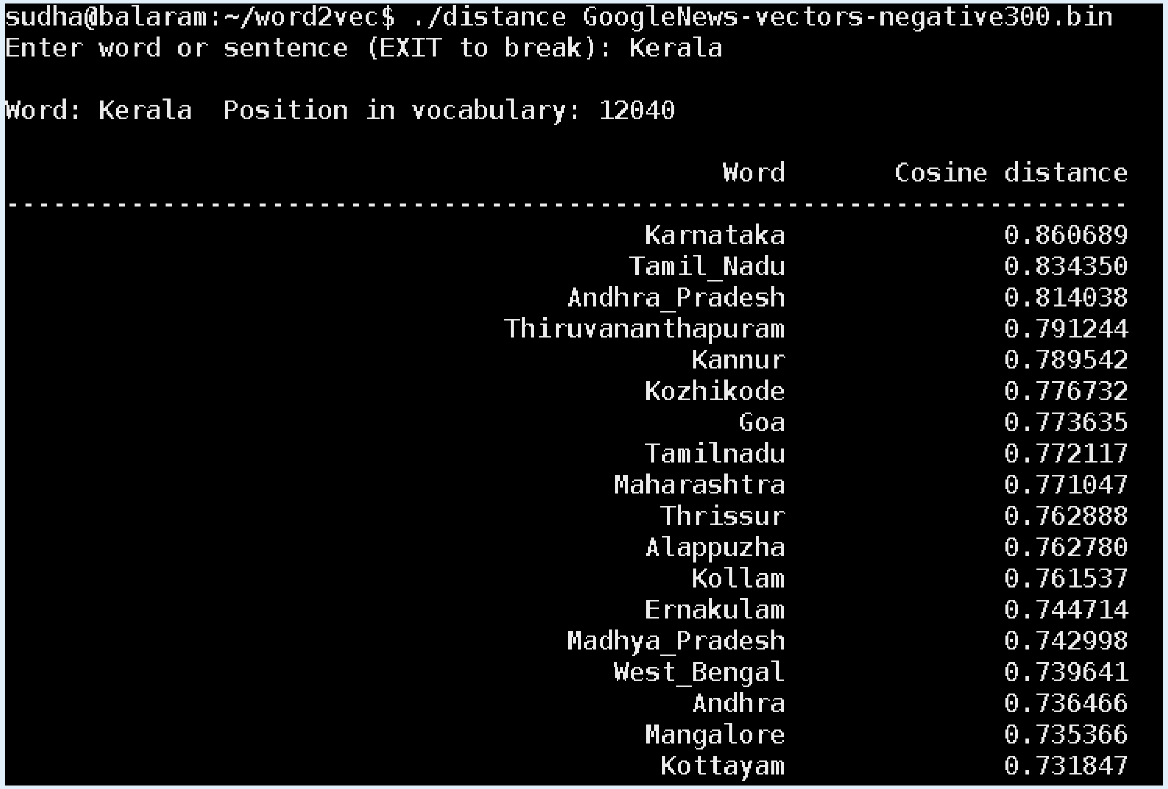
Example

```
./word2vec -train news-corpus.txt -output news_vectors.bin -cbow 1
-size 200 -window 8 -negative 25 -hs 0 -sample 1e-4 -threads 20
-binary 1 -iter 15
```

Finding related words using word2vec

- `./distance` <output_vector>

Example (related words to 'kerala')

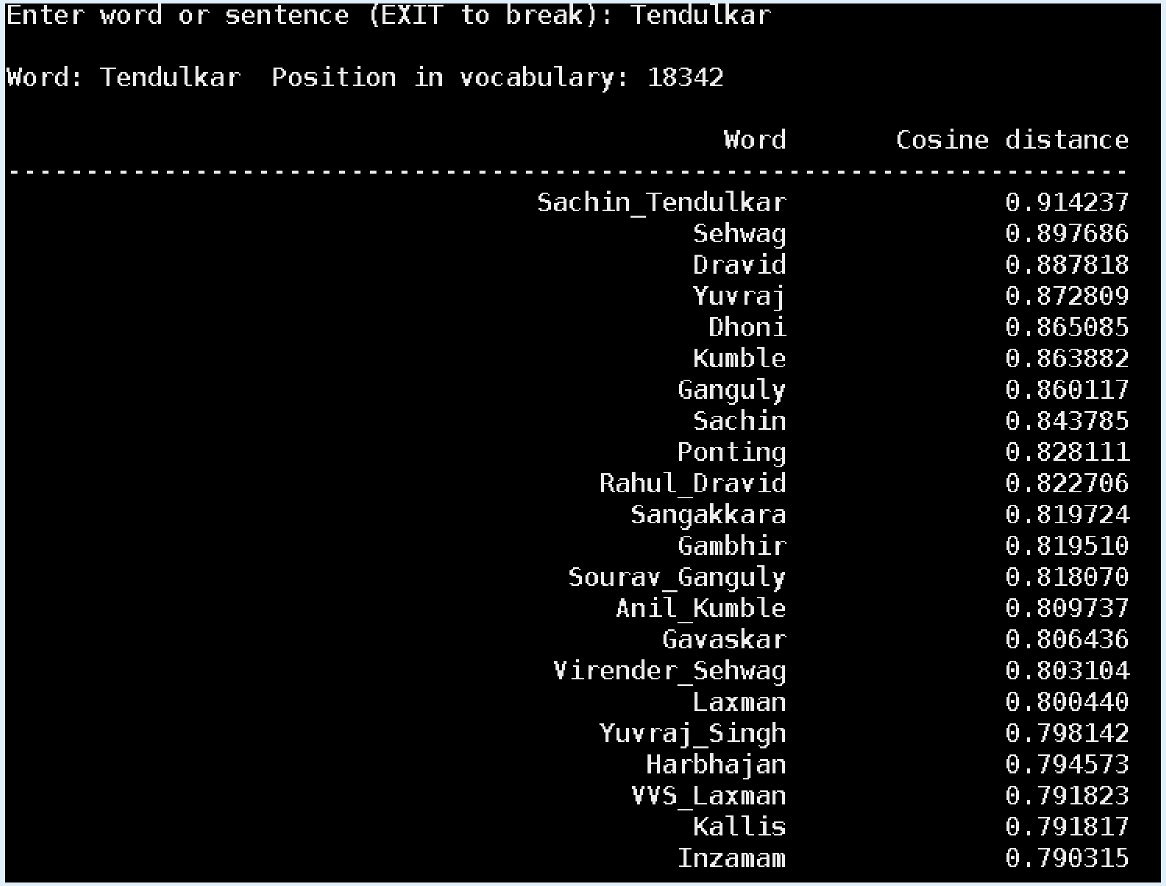


```
sudha@balaram:~/word2vec$ ./distance GoogleNews-vectors-negative300.bin
Enter word or sentence (EXIT to break): Kerala

Word: Kerala  Position in vocabulary: 12040

                                              Word       Cosine distance
------------------------------------------------------------------------
                                         Karnataka              0.860689
                                        Tamil_Nadu              0.834350
                                    Andhra_Pradesh              0.814038
                                Thiruvananthapuram              0.791244
                                            Kannur              0.789542
                                         Kozhikode              0.776732
                                               Goa              0.773635
                                         Tamilnadu              0.772117
                                       Maharashtra              0.771047
                                          Thrissur              0.762888
                                         Alappuzha              0.762780
                                            Kollam              0.761537
                                         Ernakulam              0.744714
                                    Madhya_Pradesh              0.742998
                                       West_Bengal              0.739641
                                            Andhra              0.736466
                                         Mangalore              0.735366
                                          Kottayam              0.731847
```

Finding related words using word2vec contd..

- ./distance <output_vector>

Example (related words to 'Tendulkar')

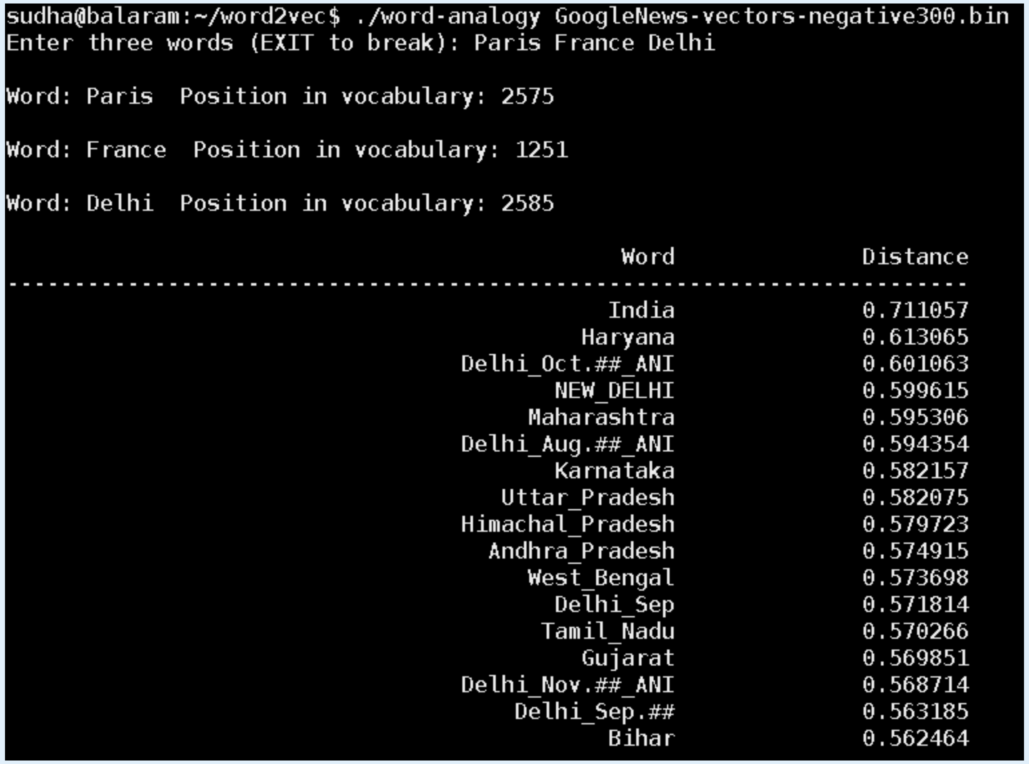


```
Enter word or sentence (EXIT to break): Tendulkar

Word: Tendulkar  Position in vocabulary: 18342

                                              Word       Cosine distance
------------------------------------------------------------------------
                                 Sachin_Tendulkar              0.914237
                                           Sehwag              0.897686
                                           Dravid              0.887818
                                           Yuvraj              0.872809
                                            Dhoni              0.865085
                                           Kumble              0.863882
                                          Ganguly              0.860117
                                           Sachin              0.843785
                                          Ponting              0.828111
                                     Rahul_Dravid              0.822706
                                       Sangakkara              0.819724
                                          Gambhir              0.819510
                                   Sourav_Ganguly              0.818070
                                      Anil_Kumble              0.809737
                                         Gavaskar              0.806436
                                  Virender_Sehwag              0.803104
                                           Laxman              0.800440
                                     Yuvraj_Singh              0.798142
                                        Harbhajan              0.794573
                                       VVS_Laxman              0.791823
                                           Kallis              0.791817
                                          Inzamam              0.790315
```

Word Analogy: Interesting properties of word2vec

- `./word-analogy` <output_vector>
  - analogy task, e.g. Paris France, Delhi ?

Example ('Paris' 'France', 'Delhi' ?)

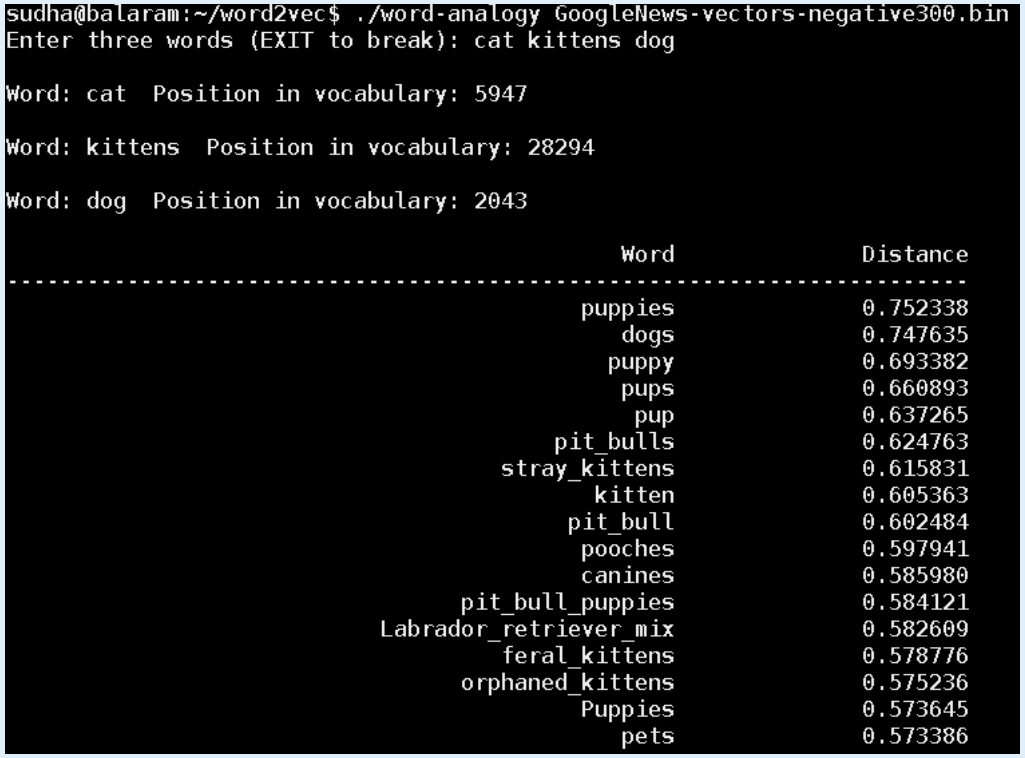


```
sudha@balaram:~/word2vec$ ./word-analogy GoogleNews-vectors-negative300.bin
Enter three words (EXIT to break): Paris France Delhi

Word: Paris  Position in vocabulary: 2575

Word: France  Position in vocabulary: 1251

Word: Delhi  Position in vocabulary: 2585

                                              Word              Distance
------------------------------------------------------------------------
                                             India              0.711057
                                           Haryana              0.613065
                                   Delhi_Oct.##_ANI              0.601063
                                         NEW_DELHI              0.599615
                                       Maharashtra              0.595306
                                   Delhi_Aug.##_ANI              0.594354
                                         Karnataka              0.582157
                                     Uttar_Pradesh              0.582075
                                  Himachal_Pradesh              0.579723
                                    Andhra_Pradesh              0.574915
                                       West_Bengal              0.573698
                                         Delhi_Sep              0.571814
                                        Tamil_Nadu              0.570266
                                           Gujarat              0.569851
                                   Delhi_Nov.##_ANI              0.568714
                                      Delhi_Sep.##              0.563185
                                             Bihar              0.562464
```

Word Analogy: Interesting properties of word2vec contd..

- `./word-analogy` $<$output_vector$>$
  - analogy task, e.g. cat kittens, dog ?

Example ('cat' kittens', 'dog' ?)



```
sudha@balaram:~/word2vec$ ./word-analogy GoogleNews-vectors-negative300.bin
Enter three words (EXIT to break): cat kittens dog

Word: cat  Position in vocabulary: 5947

Word: kittens  Position in vocabulary: 28294

Word: dog  Position in vocabulary: 2043

                                              Word              Distance
------------------------------------------------------------------------
                                           puppies              0.752338
                                              dogs              0.747635
                                             puppy              0.693382
                                              pups              0.660893
                                               pup              0.637265
                                         pit_bulls              0.624763
                                     stray_kittens              0.615831
                                            kitten              0.605363
                                          pit_bull              0.602484
                                           pooches              0.597941
                                           canines              0.585980
                                  pit_bull_puppies              0.584121
                            Labrador_retriever_mix              0.582609
                                     feral_kittens              0.578776
                                  orphaned_kittens              0.575236
                                           Puppies              0.573645
                                              pets              0.573386
```

word2vec using gensim

- Initialize, save and load the model

```
model = Word2Vec(sentences, size=100, window=5,
min_count=5, workers=4) #initialize

model.save(fname) #save

model = Word2Vec.load_word2vec_format('vectors.txt',
binary=False)  #load in text format

model = Word2Vec.load_word2vec_format('vectors.bin',
binary=True)  #load in binary format
```

word2vec using gensim contd..

```
>> model.most_similar(positive=['woman', 'king'],
negative=['man'])
[('queen', 0.50882536), ...]

>> model.doesnt_match("supper cereal dinner lunch".split())
'cereal'

>> model.similarity('woman', 'man')
0.73723527

>> model['computer']
array([-0.00449,-0.00310, 0.02421,..], dtype=float32)
```

doc2vec demo

Outline

Introduction: doc2vec

- Modifies the word2vec algorithm to find word representations for larger blocks of text, such as **sentences**, **paragraphs** or **entire documents**.
- doc2vec provides following architecture:
  - distributed memory (dm)
  - distributed bag of words (dbow)

doc2vec: Input

- **Input**: an iterator of `LabeledSentence objects`
  - Each object represents a single sentence and consists of two simple lists:
    - a list of words
    - a list of labels

Example

```
sentence = LabeledSentence(words=[u'some', u'words', u'here'],
labels=[u'SENT_1'])
```

- **Output**: vector representations for each word and for each label in the dataset.

doc2vec: Training

```
model = Doc2Vec(sentences) #store model to mapable files
model.save('/tmp/my_model.doc2vec') #load the model back
model_loaded = Doc2Vec.load('/tmp/my_model.doc2vec')
```

doc2vec: Finding embeddings for a sentence

- get the raw embedding for the sentence as a NumPy vector

```
print model["SENT_0"]
```

doc2vec: Finding most similar words or sentences

```
print model.most_similar("SENT_0")
[('SENT_48859', 0.2516525387763977),
 (u'paradox', 0.24025458097457886),
 (u'methodically', 0.2379375547170639),
 (u'tongued', 0.22196565568447113),
 (u'cosmetics', 0.21332012116909027),
 (u'Loos', 0.2114654779434204),
 (u'backstory', 0.2113303393125534),
 ('SENT_60862', 0.21070502698421478),
 (u'gobble', 0.20925869047641754),
 ('SENT_73365', 0.20847654342651367)]
```

sent2vec demo

Outline

Introduction: sent2vec

- Maps a pair of short text strings (e.g., sentences or query-answer pairs) to a pair of feature vectors in a continuous, low-dimensional space
- Semantic similarity between the text strings is computed as the cosine similarity between their vectors in that space.
- Performs the mapping using
  - Deep Structured Semantic Model (DSSM) (Huang et al. 2013)
  - DSSM with convolutional-pooling structure (CDSSM)(Shen et al. 2014; Gao et al. 2014).

sent2vec: Download and Run

- **Download**: `http://research.microsoft.com/en-us/downloads/731572aa-98e4-4c50-b99d-ae3f0c9562b9/`
- **Run**: sample/sent2vec/run.bat

sent2vec: Input and Output

- **Input**: /inFilename: input sentence pair file, each line is a pair of short text strings, separated by tab.
- **Output**: /outFilenamePrefix: output the similarity scores and the semantic vectors of the input sentence pairs

Example

Text1	**Text2**	**DSSM**	**CDSMM**
Red alcoholic drink	A bottle of wine	0.195318	0.108858
Red alcoholic drink	Fresh orange juice	0.152488	0.138266
Red alcoholic drink	Fresh apple juice	0.150574	0.193558
Red alcoholic drink	An English dictionary	-0.008468	0.022317
It is a dog	That must be your dog	0.605376	0.590164
It is a dog	It is a dog	0.952444	0.934719
It is a dog	It is a pig	0.28585	0.28473
Dogs are animals	They are common pets	0.452143	0.484175

Table: Sentence similarities using two models of sent2vec tool

References

- Huang, Po-Sen, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. "Learning deep structured semantic models for web search using clickthrough data." In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pp. 2333-2338. ACM, 2013.
- Gao, Jianfeng, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen. "Modeling interestingness with deep neural networks." In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 2014.
- Shen, Yelong, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. "A latent semantic model with convolutional-pooling structure for information retrieval." In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 101-110. ACM, 2014.

Q & A

THANK YOU..!